

Recommender Systems for Learning: Building User and Expert Models through Long-Term Observation of Application Use

Frank Linton, Hans-Peter Schaefer

The MITRE Corporation, Bedford MA, USA

Abstract. Information technology has recently become the medium in which much professional office work is performed. This change offers an unprecedented opportunity to observe and record exactly how that work is performed. We describe our observation and logging processes and present an overview of the results of our long-term observations of a number of users of one desktop application. We then present our method of providing individualized instruction to each user by employing a new kind of user model and a new kind of expert model. The user model is based on observing the individual's behavior in a natural environment, while the expert model is based on pooling the knowledge of numerous individuals. Individualized instructional topics are selected by comparing an individual's knowledge to the pooled knowledge of her peers.

1 Keywords

Learning, recommender system, instructional technology, student modeling, knowledge acquisition, logging, long-term observation, agent, data mining, cluster analysis, sequence analysis, instrumentation, organizational learning, OWL.

2 Introduction

Information technology (IT) has become the medium of work, and much learning in recent years has focused on IT skills. Learning often takes place outside formal training or educational classes and some of this informal learning is accomplished by the exchange of information among people with similar interests. The purpose of the research reported here is to study these informal knowledge acquisition processes and provide mechanisms to support them. The domain of interest is the use of information technology in the workplace, and the support mechanisms are based on information technology as well.

As a domain of interest, information technology skills have the advantage of being observable. For example, it is possible to observe text editing skills such as use of the outline tools. In contrast, other workplace skills such as writing skills, e.g., generating an outline, are primarily mental activities, and can only be inferred.

While some familiar workplace technologies, such as e-mail and intranets, support the processes of informal learning, we focus on a less familiar technology, recommender systems (Baudisch, 1999; Kautz, 1998; Resnick and Varian, 1997), which enable the pooling and sharing of information to support informal learning.

Recommender systems typically recommend information (URLs, netnews articles), entertainment (books, movies, restaurants), or individuals (experts). To our knowledge, OWL is the only recommender system that recommends new skills. Recommender systems generally work by classifying items or individuals into similar groups and making their recommendations based on a comparison of the item or individual to the group. For example, recommender systems might suggest ‘another book like the one you enjoyed,’ ‘another book enjoyed by people similar to you,’ or ‘another book enjoyed by the people who enjoy the books you do.’ We expect OWL to eventually use this third method, and discuss this point in more detail in the section on clustering users. Many recommender systems rely on explicit recommendations, yet when users must make recommendations explicitly, they often find it is not worth the trouble, hence OWL uses implicit recommendations, taking each use of an IT command as a vote for its utility.

One technological capability required for observing IT in use is logging users’ actions. We present details of our logger. The logged data can be analyzed for a variety of purposes. We provide a view of the data that characterizes the use of IT in the workplace. Thomas (1996, 1998) characterizes use of a text editor in a university setting.

One goal of this research is to provide individualized instruction by employing a new kind of user model and a new kind of expert model. This new user model is based on observing the individual’s behavior in a natural environment over a long period of time, while the new expert model is based on pooling the knowledge of numerous individuals. Individualized instructional topics are selected by comparing an individual’s knowledge to the pooled knowledge of her peers.

This approach is quite distinct from that of other systems, such as Microsoft’s Tip Wizard, which recommend new commands to users based on their logical equivalence to the less-efficient way a user may be performing a task, and it is much simpler than the more ambitious Office Assistant which uses Bayesian analysis to understand users’ actions and questions and provide intelligent assistance (Horvitz, et.al., 1998). It is also distinct from that of Intelligent Tutoring Systems, which typically recommend learning actions and activities based on a user’s capability to perform specific actions in a defined set of exercises.

One system with which OWL shares a certain similarity is PHelpS (Peer Help System) (Collins, Greer, Kumar, McCalla, Meagher and Tkatch, 1997). Like OWL, PHelpS builds user models by observing people as they go about their work and by noting how well they carry out specific tasks. Upon request, PHelpS uses the models to recommend an appropriate human helper for a task. One of the nice features of PHelpS is that a human helper can also provide information on those aspects of a task that are not performed on an information system. OWL and PHelpS are also similar in that they both attempt to transfer knowledge from those who display it to those who have a need for it, thereby promoting an overall increase in knowledge across the organization.

The system described here, OWL (for Organization-wide Learning), incrementally builds application users' skills in the context of the workplace. OWL acquires its model of the users by long-term observation of them as they go about their work. When a user decides to learn more about her application, OWL makes timely individualized recommendations about what to learn next, based on its observations of her and her peers regarding the functionality of the application they have found to be useful. The system is deployed on the MITRE corporate network, with a logger and tip presenter integrated into a text-editing application on each user's desktop computer, and a server where user data is stored and recommendations are computed. Evaluation to date confirms that the system is robust and does not interfere with user's task performance. While we do not yet have sufficient data to measure instructional effectiveness, it appears, nonetheless, that the recommender system technology used in OWL may be a relatively low-cost, high-value means for bootstrapping users' application knowledge. More details about recommender systems for learning and OWL versus related systems may be found in Linton, Joy, Schaefer, & Charron (2000).

In the following sections of the paper we describe the process of logging the commands of information technology users as they go about their everyday tasks in the workplace. We then present a summary of the data for an overall picture of their software usage. Next, we describe the process by which we compute individual user and expert models, and show how contrasting individual user models with the pooled expertise generates learning recommendations. We then present the interface by which users obtain these performance improvement tips. Finally, we examine other applications of this kind of user and expert modeling.

3 The Logging Process

A software application is a computer-based tool; the application can be instrumented so that details of its use can be observed, and those details of how the application is used by individuals can be logged. The recent shift from standalone to networked PC computing means that we now have the capability of logging the actions of a large population of individuals performing a variety of tasks with a particular software application for a prolonged period of time. The data that results from the logging process presented in this section can be analyzed and used for automated coaching. The data can be analyzed and synthesized to build models of current use, and models of expertise. Users can be individually coached by a module that compares individual performance to a synthesized expert's. Finally, the data can be analyzed to observe and promote the evolution of expertise over time.

From a practical standpoint, it is crucial that the logging process be reliable, unobtrusive, and frugal with resources, as observation takes place for extended periods of time (Kay and Thomas, 1995). The research reported here is based on logs of Microsoft Word users. The logger was written in Visual Basic for Applications (VBA). An install utility makes the initial installation easy, and an autoupdate feature makes the installation of new versions effortless. In general, it is difficult to implement loggers without access to the application's source code, but Cheikes, et.al. (1998) make available a tool for instrumenting UNIX applications. A similar tool is available for applications written in Java. The OWL, UNIX, and Java loggers may be obtained by contacting the first author.

In our system, each time a user issues a Word command such as Cut or Paste, the command is written to the log, together with a time stamp, and then executed. The logger, called OWL for Organization-Wide Learning, begins running when the user opens Word; it creates a separate log for each file the user edits, and when the user quits Word, it sends the logs to a server where they are periodically loaded into a database for analysis. A toolbar button, Figure 1, labeled 'OWL is ON' (or OFF) informs users of OWL's state and gives them ultimate control of whether they are logged or not.

The OWL database is composed of four main tables, one each for users, operating systems, documents, and commands. The document table is linked to the user table. Intermediate tables link the operating system table to the user table and the command table to the document table. Stored procedures load the log files into the database weekly.



Figure 1. The OWL toolbar button.

Figure 2 displays a sample OWL log. Logging begins when the user selects the FileOpen Command or opens a document by any other means. There is a log file for each open document. The log file is closed when its document is closed. Should a user play with, for example, View Toolbars, when no document is open, that activity is not logged. The first five lines of the log are also triggered by opening a file. They record general information: the logger version, the date and time stamp, and the author, followed by the platform, processor, and version of Word. At this point detailed logging begins. Each time the user enters a Word command, the logger adds a line to the log file. Each line contains a time stamp, the command name, and possibly one or more arguments. For example, the first line beginning 14:35:33 records these facts: at 14:35:33 p.m. the author used the FileOpen command to open the file entitled “Expertise Management.doc” containing 5266 characters. The log shows that the author performed some minor editing (copy, paste, etc.) and then printed the file. As with the system of Kay and Thomas (1995), the log does not record text a user enters; this omits some potentially useful information but preserves users’ privacy and makes logging more acceptable.

Logging captures a detailed record of users’ activities but the record may be sketchy for several reasons. First, an individual may also edit text on other systems without loggers, so some of their activity may not be captured. Second, a VBA-based logger besides omitting text, does not capture certain other keyboard actions such as Tab and Return, window actions such as MinimizeWindow or CloseWindow, nor does it capture certain mouse actions such as scrolling, nor, unlike Kay and Thomas (1995) does it distinguish *how* commands are entered (by menu, icon, or keyboard command). Finally, permitting user control over logging means that logging can be turned on and off at will, though the default is that OWL is on. To summarize then, the logged data is neither a census of the user’s actions, nor a random sample, but rather an arbitrary selection of them.

```
Initiated OWL 5.0f Logging at 05/11/99 14:35:33
System Identfier/Author 300
Platform = Windows 95 4.0
Processor: Pentium
Microsoft Word Version 8.0
14:35:33 FileOpen C:\My Documents\Expertise Management.doc
14:35:33 Doc size: 5,266
14:35:33 FileOpen Args= "Expertise Management.doc"
14:35:57 EditCopy
14:36:03 EditPaste
14:36:23 EditClear
14:36:33 EditCut
14:36:39 FormatBold
14:36:49 FilePrint
14:37:14 FileClose C:\My Documents\Expertise Management.doc
14:37:14 Doc size: 5,312
14:37:14 Filename: Expertise Management.doc
14:37:14 Path: C:\My Documents\
```

Figure 2. Sample OWL log.

4 Results of User Observation

This section presents an analysis of log data. First we present summary statistics of the users and their log data. Next we describe the relative frequencies of the different types of commands. We then present a table showing relative frequencies of each individual command, and a multidimensional bar chart to provide a holistic view of the data. Next, we display a log-log view of the data and note the trendline is a Zipf distribution. Fifth, we show how the total volume of data logged for an individual influences their apparent level of expertise.

The analysis presented in this section is exploratory in nature. The method we have used is Naturalistic Inquiry, which, to paraphrase Patton (1990, pp. 40-41) involves studying real-world situations as they unfold naturally in a non-manipulative, unobtrusive, and non-controlling manner, with openness to whatever emerges and a lack of predetermined constraints on outcomes. The point is to understand naturally occurring phenomena in their naturally occurring states. This data has been acquired from a set of users who were not randomly selected from the population, and the logged data is not a random sample of the users' actions. Therefore, all statistics presented are descriptive (of this data for this group), not predictive. Any generalizations inferred from this data must be treated cautiously until tested further.

4.1 The Subjects (Users)

The OWL project is currently (November 1999) logging 37 users. The majority of them are members of The MITRE Corporation's Center for Integrated Intelligence Systems. MITRE is a

federally funded not-for-profit corporation performing research in the public interest. The users include four Department Heads, thirteen Systems Engineers at six levels of responsibility, eight support staff, and a variety of other technical and professional staff. There are 14 males and 23 females. The users have been employed at MITRE from one to thirty-nine years with a median of 12 years. The data reported here was obtained for the Windows version of Microsoft Word 97. Word data was obtained from February 1999 to November 1999; the period of logging ranged from one to nine months per person. (Data has been collected continuously from early 1997, beginning with Word 6 users on Macintosh computers.)

During the time they were logged, the Word 97 users - as a group - used 163 distinct commands a total of 130,274 times. The average person used 58 (SD = 19) distinct commands in the period they were logged.

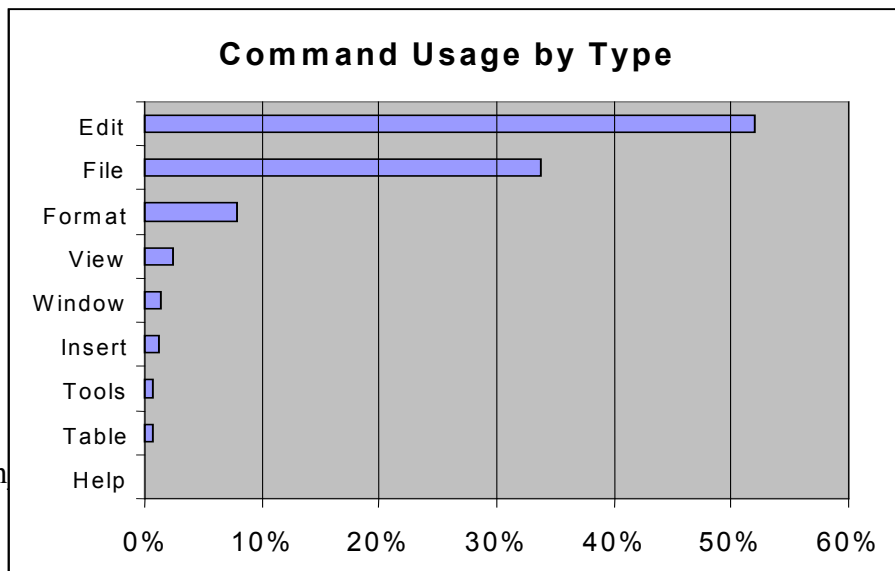
At MITRE, users are notified each time they turn on their computers that they may be monitored. However, MITRE has an organizational climate such that people do not worry that their managers might be looking at their logs, and people were not concerned about confidentiality or anonymity (based on a telephone survey by a person unconnected to the project). Most people we asked directly to volunteer did and some users recruited others for us. Those who refused either didn't use Word much or had older, slower machines. Also, for project purposes, we promised anonymity if an individual's data were to be discussed, and promised in general, to present only aggregated data. A few users asked for OWL to be removed from their machines: 1) people with older computers felt OWL slowed them down excessively, and 2) people who, in contrast to most users, frequently used some functionality of Word that invoked a known (but unresolved) OWL bug. To summarize, the positive organizational climate and low burden imposed on users were the key factors in OWL's acceptance.

4.2 Pooled Knowledge: Overall

We now turn our attention from the subjects to the commands they used, beginning with an overall, descriptive view. One of the most salient characteristics of the recorded data is the relative inequality in the use of each type of command. For example, as shown in Figure 3, the Edit commands, i.e., the commands under 'Edit' in Word's main menu, make up 52% of all commands used, while the Help commands account for only 0.03 % of the commands logged.

Figure 3. Command usage by type.

Table 1 lists the 20 most frequently occurring Word commands sequenced by their frequency of occurrence in the data, the percentage occurrence of each, and the cumulative percent of usage



for all commands up to that point in the sequence. Command names are preceded by their main menu type, e.g., File Open is the Open command in the File menu. The first two commands account for 45% of all command use, the first 10 commands account for 80%, and the first 20 commands account for 90%.

Table 1. Command sequences and percentages.

Sequence	Command	Percent	Cumulative Percent
1	Edit Delete	34.2%	34.2%
2	File Save	10.5%	44.8%
3	File Open	8.7%	53.5%
4	Edit Paste	7.9%	61.4%
5	File DocClose	5.1%	66.5%
6	Edit Copy	4.2%	70.7%
7	Format Bold	3.7%	74.4%
8	File Print	2.8%	77.2%
9	Edit Cut	2.4%	79.7%
10	File SaveAs	1.7%	81.3%
11	Edit Undo	1.5%	82.8%
12	File PrintPreview	1.3%	84.1%
13	File PrintDefault	1.2%	85.3%
14	File ClosePreview	1.1%	86.4%
15	Format Italic	0.9%	87.3%
16	Format Underline	0.8%	88.0%
17	Window DocMinimize	0.7%	88.7%
18	Format CenterPara	0.6%	89.4%
19	Edit Find	0.5%	89.9%
20	Insert Break	0.5%	90.3%

The unequal values of nominally paired command counts (for example, the log shows more FileOpen commands than FileDocClose commands) may be understood by recalling that there are multiple ways of accomplishing the same effect, that is, a file may be closed by FileClose, by FileExit, by FileSaveAs, by crashing the system; etc.

Figure 4 provides a three-dimensional view of the data. It displays the commands by relative frequency of use, frequency rank, and type. The 30 most-frequently used commands are positioned along the long axis, with the most frequently used commands at the rear of the chart. The height of each bar indicates the overall usage of each command, in percent. The first command, Edit Delete, was used 100 times more frequently than the 30th command, Format LeftJustifyText. Each vertical bar is color-coded to denote the type of command. The top 10, 20, and 30 commands account for 80%, 90%, and 94%, of usage, respectively. The Edit and File commands predominate in the first half of the list, while Format and View commands predominate in the second half. There are 6 Edit and 6 Format commands in the top 30, but Edit commands account for more than seven times the usage of Format commands. The Insert and Window commands make only one and two appearances, respectively, in the top 30 commands, and no Tools, Table, or Help commands appear.

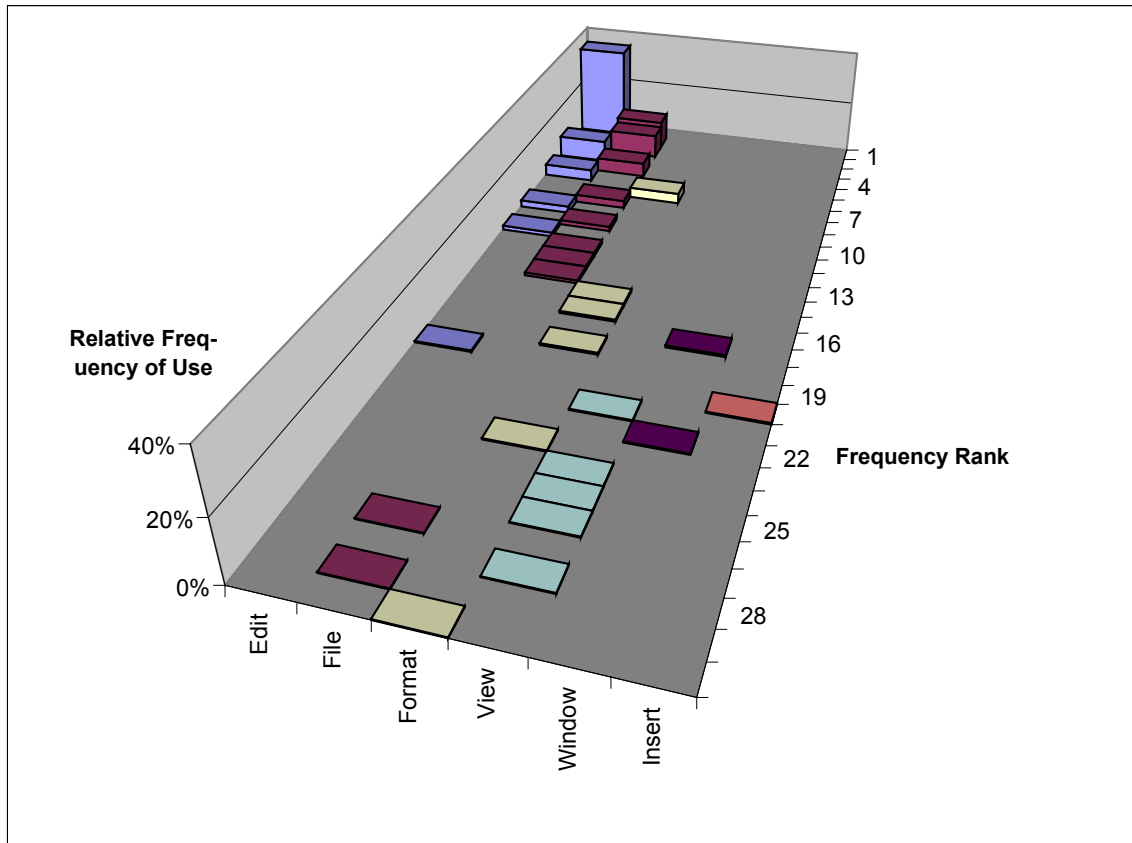


Figure 4. Comprehensive view of the data: Percentage of use by position and type.

The *Zipf distribution* appears as a straight line on a log-log plot. Many naturally-occurring phenomena such as the frequencies of word appearances in text, the distribution of city sizes, URL hit counts, etc., have Zipf distributions, and Zipf distributions have been reported for Unix commands (Thomas, 1996). The graphs in Figure 5 present command usage data for the 100 most-frequently-used Word commands. The horizontal axes present the commands ranked by frequency from 1 through 100 (the names of the first 20 of these commands were itemized in Table 1), while the vertical axes indicate their relative frequencies of use. Command usage (expressed in percent) varies by more than three orders of magnitude. Figure 5A presents the data plotted on linear scales and Figure 5B presents the same data plotted on logarithmic scales, with a linear ($y = -1.84x + 0.05$) trendline fitted to the data ($R\text{-squared} = 0.97$). The Zipf distribution of the command frequency rank is evident.

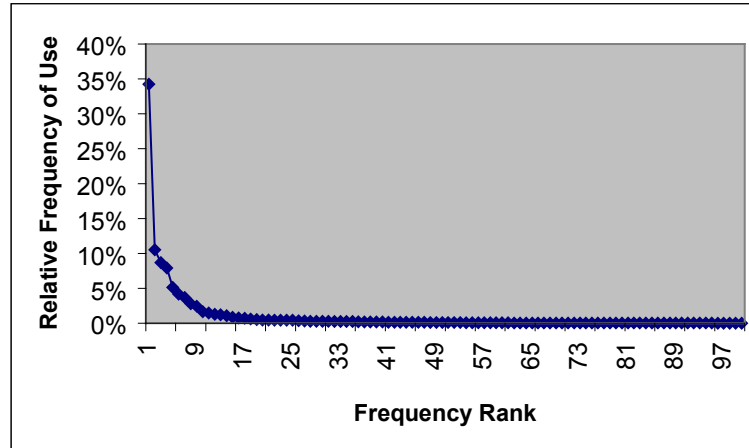


Figure 5A. Command usage data: linear scales.

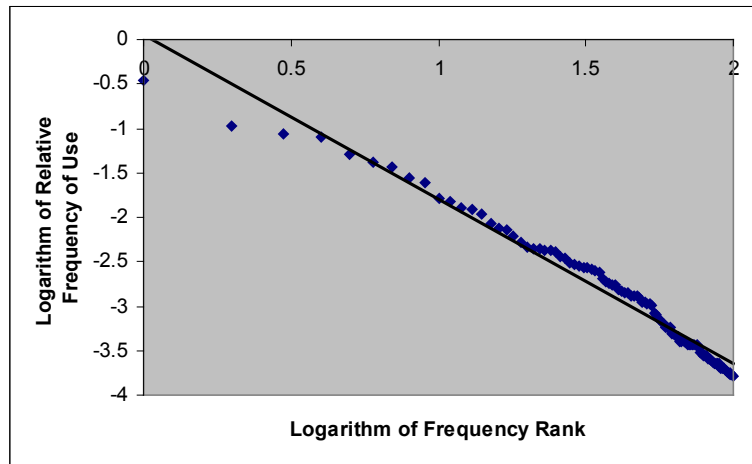


Figure 5B. Command usage data with linear trendline fitted: logarithmic scales.

4.3. Total Commands Logged Predicts Distinct Commands Logged

Once we have observed that commands are used with widely varying frequencies and that only a few commands are used frequently, we can understand that a brief observation of a user at work will not reveal her full capabilities. Instead, if a user enters, say, 400 commands per month (around average for these users), it may be several months before we see a command that occurs with a frequency of 1 in 1000, and even longer before we see the less frequently occurring

commands. In fact, if we know the relative frequencies of occurrence of commands we could predict the number of distinct commands in a user's data as a function of the amount of logged data collected on that user. Figure 6 displays the number of distinct commands as a function of total number of commands logged for 37 individuals, together with the best-fit trendline. Of course not all users know or use all commands and some users may have challenging editing tasks that require them to apply a lot of their knowledge in relatively short spurts, so we should not expect a particularly high correlation between log length and number of distinct commands. Nevertheless, log length is a remarkably good predictor ($R^2 = 0.79$) of the number of distinct commands used. That means that we should temper our conclusions about a user's knowledge of an application's functionality by considering the amount of data we have for the user.

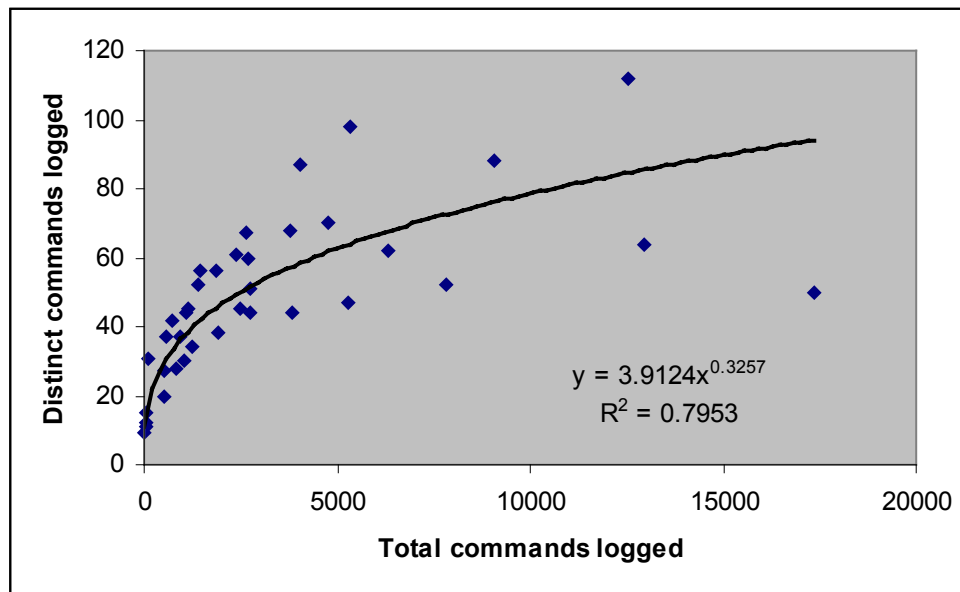


Figure 6. Number of distinct commands as a function of total number of commands logged.

We hypothesized a relationship between users' job tasks and their editing tasks, such that certain sets of users would exhibit a preference for certain sets of commands, but in the observed group of users (perhaps too small and too diverse), no such relationship was found. If such a relationship were to be found, the users should then be partitioned or clustered into subgroups which share similar usage patterns, so that recommendations to each user are based on the pooled knowledge of their peers.

Also, one might hypothesize that more-expert users would use a different subset of commands from novices, but they did not; instead they added more commands to their repertoire. If the hypothesis had been true, individual learning recommendations would have a different character, focusing on mastering a subset of commands at each level. The nature of Word usage we have observed is similar to the nature of expertise reported for high-functionality applications in Fischer (1999, p. 351) and reported for Unix and vi usage in Draper (1985). That is, "... each user is an expert - or rather a specialist - in a different corner of the system, even though the quantity of the knowledge at least as measured by the number of [distinct] commands certainly varies a lot." (p. 468) yet "... each user's knowledge overlaps other users'." (p. 467) and "... they are all in essentially the same general situation of knowing some things and being ignorant of others." (p.

468). It is because of this distributed nature of expertise that we can say that each user may contribute expertise (distinct commands) to the pooled model, while some users are more expert than others (in the sense that they use a larger number of the distinct commands).

The data collected by the OWL project covers a moderate number of users over several years, as the users went about their actual job tasks. The data is a resource for the Machine Learning for User Modeling (ML4UM) community. A subset of the data, 70,000 rows, has been posted on the ML4UM website (Linton, 1999) where anyone interested in machine learning for user modeling may access and download it for analysis. The data consists of several fields: user id, Word version, file size, file date, operating system, command name, command date, and command time. Machine learning is an active area of research, and machine learning is a promising technique for user modeling. To date (November 1999), the OWL user data is one of the few, perhaps the only, datasets publicly available for this purpose.

The usage characteristics of desktop applications described in this section may be of value to software designers, user interface designers, instructional designers, and others. We acquire this information in the process of developing automated instructional design and delivery systems using recommender systems for learning.

5 Individualized Recommendations

In this section we describe how we determine, for each individual, which IT functionality is most worth learning at the current time. We begin with a scenario that describes OWL's functionality with a hypothetical user: The typical OWL user is hardly aware that she is using OWL as she proceeds about her normal job tasks. In the background OWL records and logs the commands of the application she is using; and she can turn OWL off with a mouse click if the need should arise (Figure 1). Whenever she exits the application, OWL sends her logs to a server where they are periodically loaded into a database. From the data, OWL computes tips for each individual. The tips are stored in tip files on the server. Whenever the user decides to learn something more about the application she may click on the OWL Tips button, the current tip file is then retrieved from the server and presented to the user (Figure 8). She reviews the tips and may involuntarily emit an "Aha!" upon seeing one of them. She clicks on that tip and reviews the Word Help for the command (OWL logs this information as well). The command provides functionality that the user may have felt "Had to be there." but had been unable to find (Fischer, 1999). She tries out the command a few times and later begins to employ it. Since she installed OWL she has slowly but consistently increased her knowledge of and satisfaction with the application. This section describes how OWL computes these individualized recommendations and presents them to users.

5.1 Computing Individualized Recommendations

Our user model is a simple one. It is the list of distinct commands each person has logged, together with their respective frequencies of use. Our expert models are equally simple; they are the commands and frequencies each person would have employed if her behavior were consistent with the pooled knowledge of her peers. By comparing an individual's actual and expert models we can determine whether a particular command is not used, underused, used as expected, or overused, and make the appropriate learning recommendation.

We illustrate the recommendation generation algorithm with the following analysis. For ease of illustration, we have limited our analysis to the commands appearing under Word's Edit sub-menu. OWL performs a similar analysis using all commands. The first of the three tables in Figure 7 presents data for a selected subset (9/37) of the users. The size of the subset was

selected so as to provide a sample large enough to illustrate the recommendation generation process yet small enough to produce a legible table. In the table, each column contains data for one user and each row contains data for one command (Edit commands that were not used have been omitted). A cell then, contains the count of the number of times the individual has used the command. The columns have been sorted so that the person using the most commands is on the left and the person using the fewest is on the right. Similarly, the rows have been sorted so that the most frequently used command is in the top row and the least frequently used command is in the bottom row. Consequently the cells with the largest values are in the upper left corner and those with the smallest values are in the lower right corner.

In the middle table in Figure 7 the data have been smoothed. The *observed* value in each cell has been replaced by an *expected* value, the most likely value for the cell, using a method based on the row, column and grand totals for the table (Howell, 1982, pp. 97-101). For each cell the expected value is computed as follows

$$(\text{row total/grand total}) \times (\text{column total/grand total}) \times \text{grand total}$$

From the users' perspective, their counts of logged commands are redistributed among all the commands in the proportion that they are used by the entire group, that is, in the proportions of the Row Total column. From the perspective of a command, the total count for that command is redistributed among the users in the proportion that they contribute data to the Grand Total. For example, if an individual had recorded 500 uses of 5 edit commands and the group had used 10 edit commands in some way, then the 500 uses would be distributed over the 10 commands in the same proportion as they were used by the whole group. In computing expected values, "smoothing the data," the Row, Column, and Grand Totals remain unchanged (except for minor round off errors).

These expected values are a new kind of *expert model*, one that is unique to each individual and one that evolves over time as the individuals involved use their software to accomplish their tasks in the manner they see fit. Since the expected value in each cell reflects row, column, and grand totals, any change in value of the raw data in any cell has some influence on the computation of expected values for every cell. The reason for differences between observed and expected values, between one's actual and expert models, might have several explanations such as the individual's tasks, preferences, experiences, or hardware, but for the purpose of making a recommendation, we assume the difference indicates the lack of knowledge or skill found in the user's peers.

In the bottom table in Figure 7, each cell contains one of five symbols. The symbols are indicators of learning opportunities that are used by the recommendation generation process. These indicators are data that, when combined with domain and curriculum knowledge, result in recommendations for learning (OWL's Tips). The five symbols are: "_" (underscore), " " (blank), "New," "More," and "Alt."

OBSERVED VALUES (Selected Users)										
Count of COMMAND	EMPNMBR									Grand
COMMAND	User36	User20	User12	User17	User27	User08	User05	User30	User07	Total
EditClear	13433	17	894	827	765	73	182	208	5	16404
EditPaste	488	1555	334	51	38	192	106	3	6	2773
EditCopy	274	947	310	46	42	51	36	2	1	1709
EditCut	107	961	95	16	41	153	60	2	5	1440
EditUndo	53	415	21	94	7	50		3		643
EditFind	5	94	103	23		12	6	2		245
EditSelectAll	2	103	6	15	1	6	1			134
EditRedo	1	40		1		1				43
EditReplace		20				1	6			27
EditObject						12				12
EditPasteSpecial	1	3	1	5						10
EditGoTo		1		6		1	2			10
EditDeleteBackWord		1		8						9
EditPasteFormat						6				6
EditDeleteWord				4						4
Grand Total	14364	4157	1764	1096	894	558	399	220	17	23469

EXPECTED VALUES										
Count of COMMAND	EMPNMBR									
COMMAND	User36	User20	User12	User17	User27	User08	User05	User30	User07	
EditClear	10040	2906	1233	766	625	390	279	154	12	
EditPaste	1697	491	208	129	106	66	47	26	2	
EditCopy	1046	303	128	80	65	41	29	16	1	
EditCut	881	255	108	67	55	34	24	13	1	
EditUndo	394	114	48	30	24	15	11	6	0	
EditFind	150	43	18	11	9	6	4	2	0	
EditSelectAll	82	24	10	6	5	3	2	1	0	
EditRedo	26	8	3	2	2	1	1	0	0	
EditReplace	17	5	2	1	1	1	0	0	0	
EditObject	7	2	1	1	0	0	0	0	0	
EditPasteSpecial	6	2	1	0	0	0	0	0	0	
EditGoTo	6	2	1	0	0	0	0	0	0	
EditDeleteBackWord	6	2	1	0	0	0	0	0	0	
EditPasteFormat	4	1	0	0	0	0	0	0	0	
EditDeleteWord	2	1	0	0	0	0	0	0	0	

INSTRUCTIONAL INDICATORS										
Count of COMMAND	EMPNMBR									
COMMAND	User36	User20	User12	User17	User27	User08	User05	User30	User07	
EditClear	--	More	--	--	--	More	--	--	--	
EditPaste	--	--	--	--	--	--	--	More	--	
EditCopy	--	--	--	--	--	--	--	More	--	
EditCut	More	--	--	More	--	Alt	--	More	Alt	
EditUndo	More	--	--	--	--	--	New	--	--	
EditFind	More	--	Alt	--	New	--	--	--	--	
EditSelectAll	More	Alt	--	--	More	--	--	New	--	
EditRedo	More	Alt	New	--	New	--	New	--	--	
EditReplace	New	Alt	New	New	New	--	--	--	--	
EditObject	New	New	New	New	--	--	--	--	--	
EditPasteSpecial	More	--	--	--	--	--	--	--	--	
EditGoTo	New	--	New	--	--	--	--	--	--	
EditDeleteBackWord	New	--	New	--	--	--	--	--	--	
EditPasteFormat	New	New	--	--	--	--	--	--	--	
EditDeleteWord	New	New	--	--	--	--	--	--	--	

Figure 7. Computing Tutorial Intervention

A command whose expected value is zero need not be learned, and can be ignored; its indicator is a blank. A command that has an expected value, but is one the individual has never used, is a command the individual probably would find useful if she were to learn it; its indicator is “New.” A command whose usage is within normal range of the expected value can also be ignored. In this example, the normal range was determined empirically so as to limit the amount of recommended learning to roughly two New commands, one More command, and one Alt command per person. A similar process is used when generating the actual recommendations. In the current implementation, if a command is used more than $\frac{1}{4}$ of the expected value and less than 4 times the expected value, its use is considered within normal range. These values were set arbitrarily and can be revised when justified. The indicator for a command within normal range of the expected value is an underscore. A command that is used less than expected may be a component of tasks unknown but potentially valuable to the user; its indicator is “More.” A command that is used more than expected may indicate ignorance of more powerful ways of accomplishing some tasks; its indicator is “Alt.” For example, one user hits FileSave so frequently that it accounts for more than half of her command usage. She could instead set the SaveAutoRecoverInfo to run very frequently. For many commands, this example notwithstanding, we do find it difficult to suggest more powerful alternatives.

Notice that expected values are in some sense, an average, but OWL does not push the learner towards the skills of the average user. Instead, OWL pushes the learner towards the pooled knowledge of all users. Even the most knowledgeable individual can usually learn something from the pooled knowledge of the group. In Figure 7, for example, users 8, 17, and 20 each use 12 of the 15 commands. Users 17 and 20 both received two New recommendations.

A given volume of logged data will provide more reliable estimates of the user’s knowledge of the more frequently used commands than of the less frequently used ones. For the less frequently used commands we must do additional analysis. The correlation between volume of logged data and number of distinct commands used (Figure 6) indicates we must be careful not to equate our non-observation of a command with a lack of knowledge of that command by the user. It may be that we have not yet acquired enough data to observe it.

For example, the fiftieth most frequently used command accounts for about one-tenth of one percent of command usage, i.e., in the pooled data, it occurs once in every 1000 commands. We can use the binomial probability formula (Triola, 1983, pp. 121-129) to compute the chances of seeing the command once or more in a sample of the logged data.

$$P(x) = (n!/((n-x)!x!))(p^x)(q^{n-x})$$

In this formula $P(x)$ is the probability of exactly x successes among n trials, p is the probability of success and q is the probability of failure.

If we select a sample of 1000 commands from the data, we compute, by applying this formula repeatedly, that there is a 63 percent chance of seeing the desired command, which accounts for one-tenth of one percent of usage, once or more. If we stretch our assumptions a bit, then we can say that if we have logged 1000 commands for an individual and the individual knows a command that appears about one-tenth of one percent of the time, there is a 63 percent chance of it appearing in the log, and a 37 percent chance of it not appearing. Thus the fact that a command does not appear in the log does not necessarily mean that the user does not know the command.

The indicators mentioned above are based on current log data, but as we now see, there may not be enough data in a user’s log to determine with confidence that a command is not known to

the user. For example, let us take User 17 who has two New indicators in the example, for EditReplace, and EditObject. User 17 has logged about 2000 commands. EditReplace is used about 0.001 percent of the time, and EditObject is used about 0.0001 percent of the time. When we calculate, again using the binomial probability formula, the likelihood of seeing these commands in User 17's log once or more, we obtain figures of 86% and 18% respectively. Thus we can be reasonably sure we would have seen EditReplace if User17 knew that command, but we are much less certain about EditObject. We can use the results of these computations to cut off recommendations for which we are not reasonably sure of the user's knowledge. In any case, we should design the Tip Presenter in such a manner that OWL not insult the user if she should already be familiar with a recommendation.

OWL currently recomputes recommendations monthly. Our average user logs about 400 commands per month. Our intention is to present new sets of recommendations as frequently as possible, but only if the new set of tips is significantly different from the previous one, i.e., after collecting sufficient data to observe changes in either the individuals' or the group's use of the software. One way to automate this process might be to recompute each individual's tips weekly, but make them available to the user only when they have changed appreciably.

To review, OWL compares actual and expected values to develop instructional indicators for each command for each person each month. The expected values are those that would appear if the user were behaving as her peers - taken in the aggregate - do. Differences between the user's actual and expected values by a factor of 4 or more are taken as instructional indicators, i.e., indicators that the user might not know something her peers find sufficiently valuable to utilize with a certain frequency. OWL uses the command name, its frequency of occurrence, and the instructional indicator to make the learning recommendations it presents to the user in the form of Tips. The decision as to which commands to learn, or to learn more about, is left up to the user. If OWL's timing is right, most users will have felt the desire to learn more, but not yet done so, when the Tips are presented to them. The next sub-section describes the Tip Presenter and the Skill-O-Meter.

5.2 Presenting Individualized Recommendations

As mentioned, new sets of recommendations, known as *Tips*, are computed monthly. When the new tips are computed, we send each individual an email message telling them that they have a new set of tips. The tips are computed using all the relevant data for all the users. The "top ten" tips are the first ten tips encountered when going down the list of commands sorted according to their frequency of use. The relative ranking of the tips is determined by giving each user's first tip a rank of 100, the tenth tip a rank of 50, and the tips in the middle ranks that are proportional to their spacing in the frequency of use list. The tip file, containing the top ten tips for each individual, resides on the server. Whenever the person decides they want to learn something more about Word, they click on the Tip button on the OWL toolbar (Figure 1). The Tip Viewer (Figure 8) then retrieves the individual's tip file and displays it to them. The Tip Viewer presents the tip name, a brief description, and a relative ranking of the top ten tips. Users can then click on any tip to learn more about the command. Today, the instruction is elementary: clicking on a tip simply brings up the appropriate Word Help page. We have been field testing the OWL Tip mechanism with about 20 users for several months and are now (November 1999) scaling up with the goal of having about 100 users for the remainder of the research project.

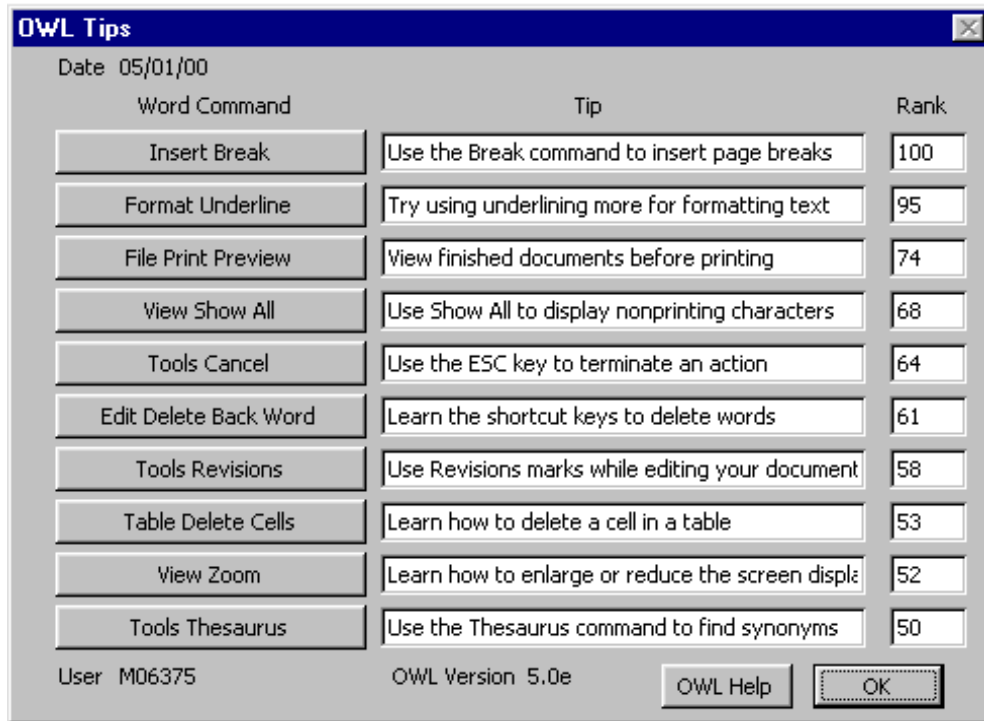


Figure 8. Image of tip presenter

OWL Help pages (as distinct from the Word Help pages that OWL points to) are also accessible from the Tip Window. OWL Help provides a list of contacts, installation instructions, a description of system functionality, a description of how recommender systems (including OWL) work, and known bugs.

6 Refining the Recommendations

We believe the current design of OWL produces good recommendations. However, we believe these recommendations could be improved by clustering users, normalizing user data, and finding meaningful sequences of commands.

6.1 Clustering Users

Cluster analysis is the process of placing individual items into groups based on the similarity of their attributes. For OWL we place individual users into groups based on similarities in their use (in percent) of Word commands. We assume that the groups will represent sets of individuals who use Word in similar ways and that recommendations based on comparison with others in their cluster will be more valuable than recommendations based on all users. We have performed a preliminary cluster analysis of the 21 individuals for whom we have more than 1000 data points. The results were unexpected.

We used the Nearest Neighbor algorithm (Fukunaga, 1990) for cluster analysis. With this algorithm, each user is considered on multiple dimensions, one for each Word command. Each user is compared to all the others on all dimensions and the two “nearest neighbors” are grouped into a single cluster. The dimensions of the new cluster are then recomputed by taking the means

of its members on each dimension. The process is repeated until there is only one cluster and then the clustering results are reviewed to find natural stopping points.

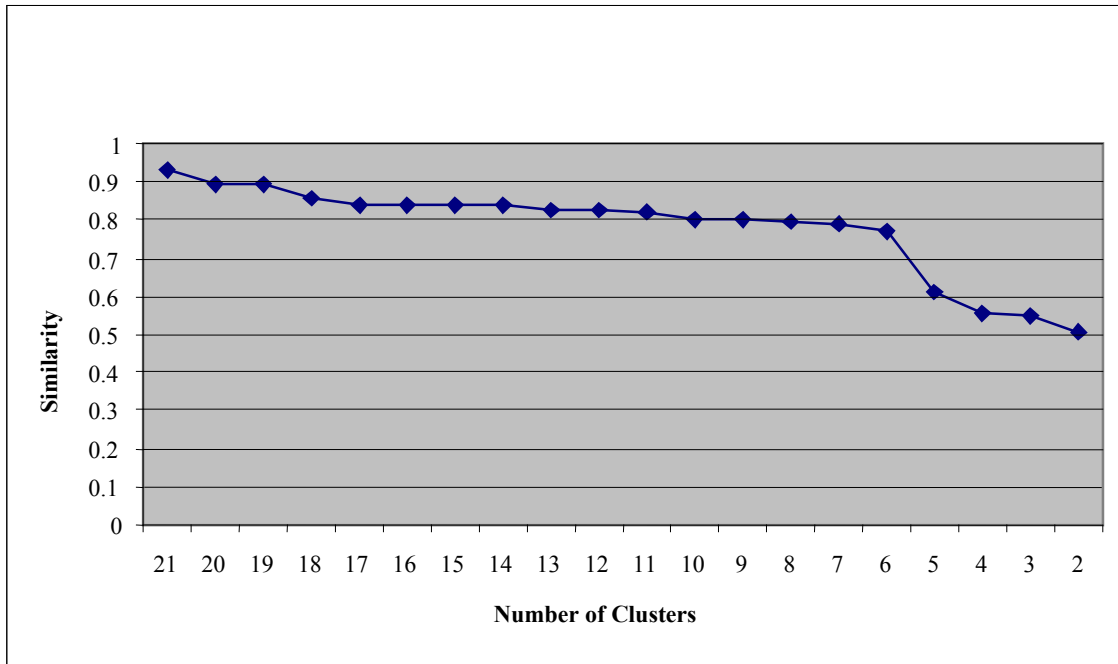


Figure 9. Maximum similarity of remaining clusters.

One natural stopping point is located where each of the remaining individuals or sub-clusters is relatively distinct from all the others. In Figure 9 we present data from the process of clustering 21 individuals by computing their nearest neighbors. The figure shows the similarity of the two nearest neighbors after each clustering step. In reducing the clusters from the 21 original clusters-of-one to just one cluster of 21, we see that the maximum similarity of the remaining clusters drops from above 0.9 to 0.5. The maximum similarity between clusters decreases slowly from 0.9 to 0.8 as the first 15 clustering operations are performed, then a big drop in similarity occurs when going from six clusters to five. In other words, when we reach five clusters, each is relatively distinct from all the others.

When two clusters have a similarity of 0.8 or more, it means that most of the commands have nearly equal percentages of use (Figure 10). For example, one of the five clusters is composed of one subcluster of seven users and one subcluster of five users, the two subclusters differ by visibly large amounts in only three of the top 20 commands.

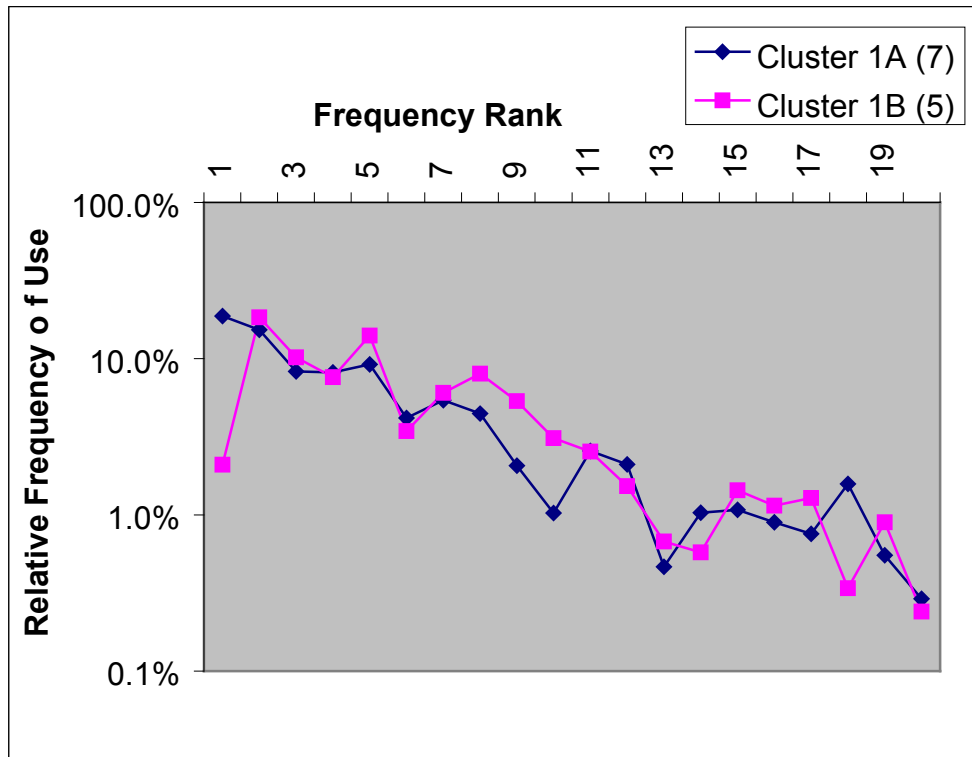


Figure 10. Two similar clusters.

Examining the five dissimilar clusters we find that one contains 12 users, one contains six users and the remaining three clusters have one user each. The cluster of 12 users is quite similar to the whole group. The cluster of six users is also similar to the whole group except for one item: their use of the delete forward key (marked “Del” on many keyboards) comprises 60% of their total usage of all commands. These six users have no demographic features in common, they have disparate ages, sexes, job titles, departments, backgrounds, and levels of experience. Similarly, the remaining three clusters of one user each are marked by excessive use of one command (a different one in each case) and by non-use of an otherwise frequently used command.

The results of cluster analysis may change to be more in line with our expectations when we increase the number of logged users, but for now it appears that cluster analysis, rather than finding groups of similar users that can be used for making focused recommendations, is instead useful for finding individuals or groups of sub-optimal users. We plan to interview these users to explore the rationales for their behaviors and possible intervention strategies, and to increase the number of logged users in hopes of finding more than one cluster of productive users.

6.2 Normalizing User Data

As we saw, OWL currently computes its recommendations based on all the commands in its database. This is not a problem if the contributions from all users are roughly equal but they are not: some users may have been logged for a longer period than others, and some are heavier users than others. Those users who have accumulated more logged data have a larger influence

on the recommendations. We might consider changing each user's contribution to a percentage, thereby giving each user equal weight, but then light users will be weighted equally with heavy users and it is in the data of the heavy users that we are likely to find the less-frequently used commands. In response to these issues of long-term versus new users and light versus heavy users we are considering *age-weighting* the data, as mentioned in Kay (1994), that is, decrementing the contribution of commands to the recommendation calculation as they age, and ignoring all commands more than one year old. This age-weighting would have the added advantage of producing recommendations that more strongly reflect users' evolving knowledge, as the latest usage patterns will get the most weight.

6.3 Finding Meaningful Sequences of Commands

We have begun to search the logged data for meaningful sequences of commands, such as Cut followed by Paste. Our hypothesis is that individuals will discover clever ways of using their software to accomplish otherwise-difficult tasks, and that we can spot these clever uses of the software (beyond single commands) by searching for sequences of commands. Two possible approaches are (1) to search for literal sequences in the log, and (2) to search for commands that occur close in time but possibly separated by other commands. Using the former method, we would spot Cut followed by Paste, but miss a cut & paste if Cut were followed by a GoTo followed by a Paste, whereas the latter method would observe both, assuming the Paste followed the Cut within the specified interval. We have decided, for simplicity's sake, to begin with the former method.

Initial analysis has revealed that most sequences are not meaningful - in the sense that a sequence represents a clever use of software that should be shared with others, and that it will take further effort to distinguish the meaningful sequences from the meaningless ones. For our initial analysis we selected a set of 15,000 commands recorded by 10 users creating 1000 files using 120 distinct commands. The 120 commands have 14,400 possible two-command sequences. Our analysis turned up about 1100 actual command pairs.

To determine whether these pairs are meaningful, one can compare their frequency of occurrence to the likelihood of their occurrence given the individual frequencies of the commands that compose them. That is, if Cut followed by Paste is a deliberate sequence, the sequence should occur significantly more often than chance would dictate, and we can compute the chance of the pair occurring based on the individual frequencies of Cut and Paste in the logged data.

In our data sample, Cut & Paste occurred 4.5 times more often than chance alone would dictate, a good sign. However, Cut followed by TableInsertColumn occurred 8 times more often than chance alone would dictate, and it is doubtful that Cut followed by TableInsertColumn is a command pair worth learning. Clearly, likelihood of occurrence alone is not sufficient to reveal useful command sequences. It may be that meaningful command sequences are correlated with individual users, individual files, files of a certain length, etc. Further exploration is required to discover an algorithm for detecting meaningful command sequences.

7 Remarks

In this section we discuss deploying OWL software, designing a Skill-O-Meter, evaluating the OWL project, and applying 'recommender systems for learning' to other application domains.

7.1 Deploying OWL

OWL’s development and deployment is an iterative process. Each version is more robust and is deployed on more users’ computers; with more users, more problems crop up. The current version, with 37 users, requires one quarter of a staff person’s time to maintain. Her activities include:

- Test new versions of the OWL logger
- Troubleshoot and fix problems that occur on users’ machines
- Monitor the OWL server – make sure it is on line
- Run various stored procedures to load logs, generate tip files, etc.
- Maintain data quality and back up the database
- Create all documentation and maintain the OWL project page
- Update the Tip table

Developing new versions of OWL usually requires some work by the database programmer and the Visual Basic for Applications programmer. Design is a team affair. Data analysis is the PI’s responsibility.

7.2 Skillometer

In order to focus on users’ knowledge instead of their ignorance, we have developed an initial design for another tool intended to show users their level of skill, the Skill-O-Meter (Figure 11). Like the skillometers of Anderson (1992) and Lesgold (1992), the Skill-O-Meter will show users what they know instead of focusing on what they don’t know. The OWL Skill-O-Meter will show users how much they know compared to the pooled knowledge of a demographic group the users select themselves, such as users in their division, department, job family, or location. As designed, the horizontal bar graphs represent the percentage of commands that the individual uses within the normal range, when compared to the selected demographic group. Clicking on a bar graph label (indicating a main menu selection) would bring up a list of commands in that menu, together with their learning recommendations.

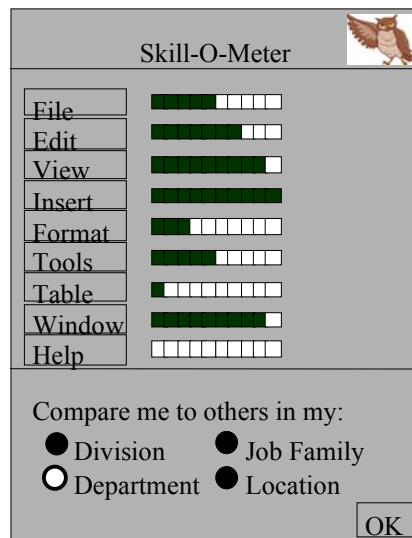


Figure 11. Image of Skill-O-Meter.

7.3 Evaluating the OWL Project

Evaluation, for a project such as OWL, consists of evaluating the results of each phase of development and deployment. We ensured the software was robust, stable, and reasonably bug-free first by laboratory testing, then by pilot testing it. During the pilot period we took the initiative and frequently contacted our users to turn up problems and resolve them. Also, we released the software in phases, first the logging module, then the file transfer module, then the feedback (OWL Tips) module. “Customer support” is always available during working hours, and we proactively check with users for problems.

We maintain the quality of the data by weekly inspections of the logs received from users, the data added to the database from the logs, and (monthly) the tips generated, for unexpected events, and if found, trace their causes and make corrections.

We have only recently begun to compute tips and make them available to users beyond the pilot group, thus we have not yet calculated effectiveness, learning, or knowledge gain, but we are collecting the information that we require to make these calculations.

To measure the effectiveness of OWL Tips we log the use of the Tip Window and the selection of individual tips. Thus we know whether the user has looked at the Tip Window and which of the individual tips she has chosen to investigate further. The standard command logging process reveals whether the user tries out a command after seeing a tip recommending it. To measure learning we will look for the continued use of a command that was introduced by a tip.

To measure knowledge gained by individuals we could compute the number of commands used divided by the expected number for a given log length (Figure 6), and observe how this ratio changes over time. For example, referring to Figure 6, one user has logged about 8000 commands. According to the trendline, we would expect this person to have used about 72 distinct commands, but they have used about 52, for a current score of 52/72 of their expected ‘knowledge’. By the time a person has logged 10,000 commands we would expect, according to Figure 6, for them to have used 80 distinct commands. If OWL were an effective coach, perhaps by the time this person has logged 10,000 commands she will have begun to use 80 commands at their normal frequencies. As it stands, this analysis weights each command equally. The user will have a score of 100 percent if she uses *any* 80 commands. One refinement might be to weight the commands according to their position in the frequency of occurrence list.

Thomas (1996, p. 81) found the type-token ratio (i.e., the number of distinct commands per unit of logged data) to be stable for students and for experts, yet he also found experts to be more knowledgeable than students, and acknowledges that there must be some means by which students may become experts. Our hypothesis is that OWL’s Tip recommendations could serve as a mechanism for augmenting the natural means by which less knowledgeable users become more knowledgeable users.

Knowledge gained by the organization (versus that gained by the individual) may be one of the more important outcomes of using OWL, or any training system, from a cost/benefit perspective. One way to measure organizational knowledge gained is by computing the total number of users of each command. Except for the most-frequently used commands, it should not be the goal that every user employ every command, but instead that every user have the opportunity to consider the added value each new command might provide him or her. OWL will ensure the learning opportunity. Users must make their own judgements about the value of the proffered knowledge. One way to measure the organizational knowledge gained, then, is to measure the increase in users per command, or number of tips taken up, with a weighting scheme

that favors the more frequently used commands. In doing so, evaluators must keep in mind that “everyone uses all commands” is not the target.

Kirkpatrick’s (1975) four questions regarding the evaluation of training are: Did the trainees like the experience? Did they learn the skills? Do they use the skills? And, does it matter (in terms of organizational effectiveness)? Most training evaluations answer only the first question. OWL can provide quantitative answers to the second and third questions, and data for inferring the fourth.

7.4 Further applications of recommender systems for learning.

In this paper we have presented a recommender system for learning and our description has focused on one specific means of applying such a recommender system, namely to recommend new application commands to individual users. However, the technique we have applied in OWL could also be used to recommend data sources such as URLs, or to recommend new functionality, or classes, in a programming language such as Java. For example, we applied our methodology to recommend URLs from the MITRE corporate intranet to a group of users. For analysis, we selected a group of managers from one division and the URLs they had visited in the previous 6-month span. 175 of the URLs had been visited by two or more managers two or more times. We limited our analysis to those 175 URLs. As with OWL, we computed the expected values from the observed values by smoothing the data, and computed the potential recommendations for each manager by comparing the observed values and the expected values. If we expected the manager to visit the URL once or more, and she had not visited the URL (in the last six months), we included the URL, otherwise not. Finally, we determined an individualized set of recommendations for each manager by taking the ‘Top 10,’ the 10 unvisited URLs most-visited by other managers. The recommendations appeared reasonable and we are implementing a browser-based tool that will enable any MITRE employee to visit the intranet URLs his or her peers have found valuable.

Yet another domain for the approach taken in OWL, besides application functionality and data sources, is modern object-oriented programming languages such as Java. Java has more than 1500 classes and at least 10 times that many methods. Consequently the initial effort to master enough of the language to program effectively is significant. A database of the classes and methods found most-useful by the programmers in one’s organization could form the basis of a curriculum. Such a database could be developed by parsing the organization’s files of Java code and noting the frequencies with which Java’s classes and methods were used. Individual learning recommendations would be computed as in OWL.

8 Summary of Contributions

To summarize, not only has information technology become the medium in which much work is performed, information technology skills have become a significant portion of workers’ knowledge. In contrast to other tasks, information technology tasks are observable, and can be logged and analyzed for various purposes. We have presented the results of a long term study of how one information technology application, Microsoft Word, is used. We have shown that the frequency of command usage for Word follows the Zipf distribution. We have also analyzed information technology usage for the purpose of constructing individual user models based on long term observation of users in their natural environment and on building expert models for each user based on pooling the knowledge of several users. We have shown how we create

individualized instruction based on comparing the knowledge of each individual to the pooled knowledge of her peers.

We have made logged data from the OWL project available to researchers. We have also made the logging software itself available to researchers.

We have applied the concept of a recommender system to help users decide what to learn next. We determine the expected use of each command by each individual: an individualized expert model; and take large deviations from this model as indicators of ineffective software use -- and learning opportunities. We described two methods, one planned and one implemented, of communicating these learning opportunities to users. We outlined how we intend to refine our analysis, and made the claim that using recommender systems for learning is a general technique applicable not only to desktop applications, but also to data sources and programming languages as well.

Acknowledgements

This research was funded by The MITRE Corporation research grant 51MSR80E. The authors wish to thank Deborah Joy for her dedication to ensure data quality, Dawn Hersey for the database work, Carsten Oertel who performed the cluster analysis, Andy Charron who developed an early version of OWL, the anonymous reviewers, and all the OWL users. An abridged version of this article appeared in UM99: User Modeling: Proceedings of the Seventh International Conference (Linton, Joy, & Schaefer, 1999).

References

- Anderson, J., Corbett, A., Fincham, J., Hoffman, D., & Pelletire, R. (1992). General Principles for an Intelligent Tutoring Architecture. In J. Regian and V. Shute (Eds.), *Cognitive Approaches to Automated Instruction* (pp. 81-106).. Mahwah NJ: Erlbaum.
- Baudisch, P. (1999). CHI' 99 Workshop: Interacting with Recommender Systems, 15/16 May 1999. Part of the CHI '99 Conference, Pittsburgh, Pennsylvania, USA. <http://www.darmstadt.gmd.de/rec99/schedule.html>
- Cheikes, B., Geier, M., Hyland, R., Linton, F., Rodi, L., and Schaefer, H. (1998). Embedded Training for Complex Information Systems. In *Proceedings of ITS 98*. Springer-Verlag.
- Collins, J., Greer, J., Kumar, V., McCalla, G., Meagher, P., and Tkatch, R. (1997). Inspectable User Models for Just-In-Time Workplace Training. In A. Jameson, C. Paris, and C. Tasso (Eds.). *User Modeling: Proceedings of the Sixth International Conference, UM97* (pp. 327-337). New York: Springer Verlag.
- Draper, S. (1984). The Nature of Expertise in UNIX. In B. Shackel (Ed.), *Proceedings of Human-Computer Interaction - INTERACT '84* (pp. 465-471). Elsevier Science Publishers B. V. (North-Holland).
- Fischer, G. (1999). User Modeling: The Long and Winding Road. In J. Kay (Ed.), *UM99: User Modeling: Proceedings of the Seventh International Conference* (pp. 349-355). New York: Springer Verlag.
- Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition*. San Diego: Academic Press.

- Horvitz, E., Breese, J., Heckerman, D., Hovel, D., and Rommelse, K. (1998). The Lumiere Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, Madison, WI, July 1998* (pp. 256-265). Morgan Kaufmann: San Francisco.
- Howell, D. (1982). *Statistical methods for psychology*. Boston: Duxbury Press.
- Kautz, H. (1998). *Recommender Systems: Papers from the AAAI Workshop, Henry Kautz, Chair, July 27, 1998, Madison Wisconsin. Technical Report WS-98-08.* <http://www.aaai.org/Press/Reports/Workshops/WS-98-08.html>
- Kay, J. (1994). Lies, damned lies and stereotypes: pragmatic approximations of users. In B. Goodman, A. Kobsa, and D. Litman (Eds.), *Proceedings of UM94 - 1994 User Modeling Conference* (pp. 175-184). Boston: User Modeling Inc.
- Kay, J., and Thomas, R. (1995). Studying long-term system use; computers, end-user training and learning. *Communications of the ACM*, 38(7), 61-69.
- Kirkpatrick, D. L. (1975). Techniques for evaluating training programs. In D. Kirkpatrick (Ed.), *Evaluating training programs* (pp. 1-17). Madison WI: American Society for Training and Development.
- Lesgold, A., Eggan, G., Katz, S., & Rao, G. (1992). Possibilities for Assessment Using Computer-Based Apprenticeship Environments. In J. Regian and V. Shute (Eds.), *Cognitive Approaches to Automated Instruction* (pp. 49-80). Mahwah NJ: Erlbaum.
- Linton, F., Joy, D., & Schaefer, H-P. (1999). Building User and Expert Models by Long-Term Observation of Application Usage. In J. Kay (Ed.), *UM99: User Modeling: Proceedings of the Seventh International Conference* (pp. 129-138). New York: Springer Verlag.
- Linton, F. (1999). Dataset: Usage of Microsoft Word Commands. A repository of 70,000 rows of data at the Machine Learning for User Modeling web site: <http://zeus.gmd.de/ml4um/>
- Linton, F., Joy, D., Schaefer, H-P., & Charron, A. (2000) OWL: A Recommender System for Organization-Wide Learning. *Educational Technology and Society*. <http://ifets.ieee.org/periodical/>
- Patton, M. (1990). *Qualitative evaluation and research methods*. 2nd. Ed. London: Sage.
- Resnick, P., and Varian, H. (1997). Introduction to Special Section on Recommender Systems. *Communications of the ACM*, 40(3), 56-58. <http://www.acm.org/cacm/MAR97/resnick.html>
- Thomas, R. (1996). Long term exploration and use of a text editor. Unpublished doctoral dissertation. University of Western Australia.
- Thomas, R. (1998). *Long term human-computer interaction: An exploratory perspective*. New York: Springer Verlag.
- Triola, M. (1983). *Elementary statistics*. 2nd. Ed. Menlo Park CA: Benjamin/Cummings.

VITAE

Dr. Frank Linton

The MITRE Corporation, 202 Burlington Road, Bedford MA 01730, USA

Dr. Frank Linton is a Lead Scientist for Instructional Technology in the Artificial Intelligence Center of The MITRE Corporation. Dr. Linton received his B.A. in Adult Training and Development from the University of Massachusetts in 1985, his Ed.M. in Educational Technology from Harvard in 1986, and his Ed.D. in Artificial Intelligence in Education from the University of Massachusetts in 1995. Dr. Linton draws on advanced information system technologies to enhance the instructional capabilities of computer mediated learning, including intelligent tutoring systems, distance learning, collaborative learning environments, and recommender systems for learning. The current work reflects his ongoing interest in tools to support informal workplace learning.

Hans-Peter Schaefer

The MITRE Corporation, 202 Burlington Road, Bedford MA 01730, USA

Mr. Schaefer is a Senior Artificial Intelligence Engineer at The MITRE Corporation. He received his B.S. in Electrical Engineering from Villanova University in 1977 and M.S. Computer Science from Rensselaer Polytechnic Institute in 1982. Mr. Schaefer has developed training systems for the Government while working for UNYSIS Corporation and provided acquisition support for Government training systems at The MITRE Corporation. In the last five years, Mr. Schaefer has conducted research in advanced computer based training technology and has focused on transferring this technology. His research has included the user of reflective-followup in training, case-based reasoning and student monitoring. Recently he has developed the software used to monitor Microsoft Word users for the OWL project.