Dynamic Detection of Novice vs. Skilled Use Without a Task Model

Amy Hurst, Scott E. Hudson, Jennifer Mankoff
Human Computer Interaction Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
{akhurst,scott.hudson,jmankoff}@cs.cmu.edu

ABSTRACT

If applications were able to detect a user's expertise, then software could automatically adapt to better match expertise. Detecting expertise is difficult because a user's skill changes as the user interacts with an application and differs across applications. This means that expertise must be sensed dynamically, continuously, and unobtrusively so as not to burden the user. We present an approach to this problem that can operate without a task model based on low-level mouse and menu data which can typically be sensed across applications at the operating systems level. We have implemented and trained a classifier that can detect "novice" or "skilled" use of an image editing program, the GNU Image Manipulation Program (GIMP), at 91% accuracy, and tested it against real use. In particular, we developed and tested a prototype application that gives the user dynamic application information that differs depending on her performance.

Author Keywords: Statistical Models, Intelligent User Interfaces

ACM Classification Keywords: H5.2 [Information interfaces and presentation]: User Interfaces - Graphical user interfaces, H1.2 [Models and Principles]: User/Machine Systems

INTRODUCTION

Not all users interact with computer systems in the same way, yet most applications act the same for all users. Expertise and skill in particular has a major impact on how a user interacts. For example, a novice must search for menu items while an skilled user typically remembers where they are located. Detection of skilled use can allow an application to better adapt itself to a user's needs. The work presented here seeks to support adaptive systems by (approximately) detecting skilled use in a dynamic and application independent fashion.

Specifically, we have built statistical models that are able to use input event stream data to classify actions as novice or skilled behavior with an accuracy of 91%. The statistical models were trained on data from a study where users performed repetitive tasks (illustrated in Figure 1), resulting in their progression from novice to skilled behavior. Using the resulting classifier, we created a prototype application that adapts to expertise based on performance.

Our statistical models can dynamically identify a user's skill level with an application simply from observing mouse and menu data, without knowing a task model. Our prototype software could operate on any GTK application, and our overall approach is generic and could be integrated into operating systems such as Windows, OS X, or Linux and used across applications via simple monitoring of the event streams provided by those systems.

We envision using knowledge about skilled use to help users by adapting the interface to meet the user's needs [2] or providing tailored intelligent help [16]. For example, dialog boxes, help systems, or automatically generated previews could be set to only appear when the user is likely to need them, and could also provide a level of detail appropriate to the user's skill.

Our approach to statistical machine learning, derived from that presented in [12], depends on a set of *features*, or metrics, that quantify an interaction based on raw data (such as mouse motion). Given a set of *training* data including labeled examples of both novice and skilled use, it is possible to (1) determine which features are most predictive of skilled use and (2) train a *classifier* that, given unlabeled test data, returns an estimate of whether the test data represents novice or skilled behavior. An effective strategy for making an accurate classifier is to generate a large set of potential features, and to use a feature selection algorithm to choose which features will best classify the data.

Based on this approach, our contributions are: (1) knowledge about what detailed features are predictive of skilled behavior, that complements what has been found in empirical studies of skilled use; (2) a classifier that can detect expertise by analyzing a single menu selection with 91% accuracy.

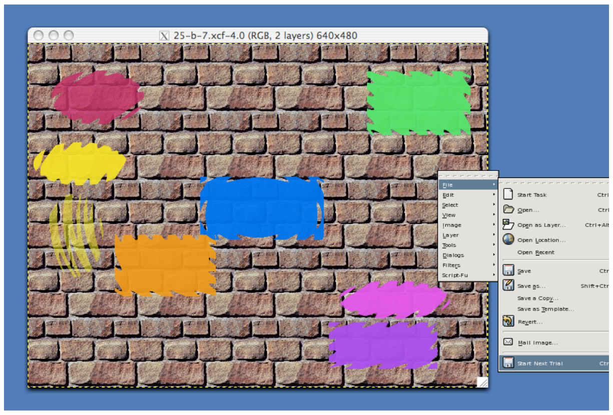


Figure 1. Our test application after several user trials. Menus in this application are similar to those appearing in many other applications.

In the following sections we describe how novice and skilled use differs and describe quantitative features of each. Next we discuss how we collected a corpus of data to build our statistical models. Finally we present a prototype application we developed that adapts help information based on classifications of expertise from our statistical model. We describe the results of a study testing the application in real use. We conclude with a summary of our findings and a discussion of how we envision deploying adaptations as future work.

UNDERSTANDING DIFFERENCES IN EXPERTISE

To successfully differentiate novice from skilled behavior it is important to isolate easily observable differences between typical actions of each. To do this we draw from research on Interaction Design and Human Factors to develop a set of features that are likely to reflect these differences quantitatively. Statistical methods from machine learning can then be used to identify which features are most able to differentiate skilled from novice use and to create predictive models based on these features.

In the literature, the terms expert behavior and skilled behavior are both terms used to describe expertise. For clarity, we will use the word "expert" to refer to mastery of an activity (such as editing a document) and the word "skill" to refer to mastery of a task (such as selecting a menu item). Where past research is non-specific, we will use the word "expertise". We make the assumption that experts are also skilled, an assumption supported by [16].

Qualitative differences

Intuitively, expertise is a mixture of knowledge, speed, and comfort with a particular piece of software. Norman's "The Design of Everyday Things" [22] provides a useful summary and conceptualization of the relevant human and cognitive factors. Following Norman, expertise can be seen as the difference between someone who knows key information and can draw on existing plans of action versus someone with less detailed knowledge who must plan more dynamically based on information in the environment. More succinctly, given a situation where a user wants to accomplish something, experts, and skilled users, tend to use *knowledge in the head*, or *recall*, to achieve this goal, whereas, novice users tend to rely on *knowledge in the world*, or *recognition.*

These differences in behavior should manifest themselves in measurable differences in user actions. It is these differences which will drive the predictive models developed here. Quantitative studies isolating these differences are one important source of ideas for developing features that can differentiate novice and skilled behavior.

Quantitative indicators

Features useful for driving statistical models can be drawn from a number of sources, but need to be predictive and easy to collect. In this work we have developed a set of features indicative of skilled behavior based on properties observable in low level user inputs (such as mouse velocity or detailed timing of common operations). This allows them to be employed dynamically and in an application independent way (*i.e.*, without prior knowledge of a task model).

One particularly fruitful area for development of these features is in observable differences in the use of common interaction techniques. By developing features at the interaction technique level, we can leverage some basic information about the nature of the user's actions. However, by focusing on the application independent aspects of generic, common interaction techniques, we can still maintain our task independence. One of the most common interactions used across applications is menu selection.

Skilled vs. Novice Use of Menus

When an expert or skilled user wishes to accomplish a task with a menu, the main activity she has to accomplish is selecting the correct menu item. Because she can typically *recall* the location of the menu item, she does not need to search for it, which allows her to select it more quickly than she might otherwise. Alternatively, she may use an even faster keyboard shortcut to perform the action.

The size and organization of a menu affects a user's selection time. The time to select a menu location has been reported as proportional to menu size, for both sorted and random menu items [23]. When a skilled user is interacting with a sorted menu, she is easily able to memorize the positional location of menu items, which allows her to select items more quickly. This is most evident with the first menu item in a sorted menu [23,26]. Hornhoff [11] extended this work by building user models to describe how a user makes a menu selection when she knows where the menu item is located.

In contrast, novices do not typically know what menu item they want to select, or where it might be located, and usually have to search for it, using clues in their environment. This affects even low level processing of menu items: for example, novice users tend to rely on word recognition while searching in menus, while experts and skilled users extract global visual features from the display (such as the number of characters in the menu item name) when searching for an item who's location is not fully recalled [13]. There is much debate in the literature regarding the exact search strategies typical of different expertise levels [4,5,10]. However, the consensus of this research is that users perform a top-to-bottom search of a menu instead of a random one when looking for a menu item.

This body of research suggests that skilled and expert users are able to remember the names of items, anticipate the location of items, and leverage visual features of items, all of which allows them to make faster selections with more skill. This suggests that features that measure speed could be predictive of expertise. It also suggests that features that can differentiate searching behavior from other types of motion may help to differentiate novice and skilled use. Example features that approximate searching including the number of submenus that are opened, and how often the cursor "dwells" over a menu item.

Skilled Performance Modeling

A source of information complementary to the empirical studies of interaction described above is the literature on human performance modeling such as Fitts' Law [9], the Steering Law [1] and the Keystroke-Level Model (KLM) [16,6]. Fitts' law describes skilled human performance in rapid aimed movements. This law applies when users are both individually clicking targets as well as dragging between targets (such as selecting a menu in the menu bar). The Steering Law is related to Fitts' Law, but it describes skilled motion that is path constrained, such as horizontal motion through a menu item to reach a second level menu. All of these models assume pure motion such as that typical of a skilled user, rather than motion mixed with browsing, searching, or other characteristics of novice interaction.

At a high level, both Fitts' Law and the Steering Law tell us that both the distance to a target and the size of a target (or the path constraint in the case of the Steering Law) affect the speed with which it is selected. Most operating systems do not report on the exact size of targets such as menus, so we were not able to directly use either law as a feature. Even so, the underlying ballistic properties of mouse motion (the velocity and acceleration curves) that result in Fitts' law could be a source of useful features.

Keystroke Level Modeling typically involves constructing a detailed, task specific model of expert behavior. Since we cannot be sure what task a user is performing, the use of typical Keystroke Level Models is problematical. However, it is still possible to employ the KLM technique for small subtasks that are easily identifiable and appear ubiquitously across applications (such as target acquisition, single or multi-level menu selection, or dragging tasks).

For example, we can use KLM to create a model that predicts the amount of time it would take to make a multi level menu selection, using operators (in italics) and predicted times taken from [6,16]. Once a menu is visible, the user would perform a *Look & Mental Operator (LM)* action to decide where the menu item of interest is (1.35 seconds); *Travel (T)* to move the mouse to the target (1.1 seconds*)*; and *Click (C)* by pressing and releasing the mouse button (0.2 seconds). A complete model including multiple submenus is *#submenus * (LM + T) + C*, or *2.45*#submenus + 0.2 seconds*. A possible feature would compare the time it took a user to perform an action to the time this equation predicts for the same action, presuming that skilled users would perform at speeds closer to the predicted time than novices would.

Mouse vs. Keyboard and Other Data

In addition to mouse and menu based features we have also considered alternatives including monitoring use of on-screen dialogs, help browsers, or keyboard logs. For example, interesting features of a user's interactions with on-screen dialogs would be the use of help browsers, or frequent canceling of dialogs. Possible features of keyboard logs include detecting actions that are performed and then immediately undone, or detecting the use of keyboard shortcuts.

Features of this sort clearly offer some promise for differentiating novice and skilled use. However we have not concentrated on their use because they present several drawbacks when compared to a variety of mouse-based features. Most importantly, they occur less frequently than mousing during typical interaction and hence would limit how dynamic our detection results could be. In addition, some of these actions, such as menu shortcuts, occur less ubiquitously (*e.g.*, for only some menus) or are rarely used by some individuals [15]. Finally, in some cases these features are invoked in mostly application specific ways (*e.g.* use of an application specific button or menu item to invoke help).

CREATING A CLASSIFIER

In this section we discuss the specifics of creating a classifier (or *predictive model*) for detecting novice vs. skilled actions from mouse and menu data. As just described, we leveraged empirical knowledge and models of human performance to create a number of promising candidate features for differentiating novice and skilled behavior. However, rather than trying to directly engineer the very best set of such features *a priori*, we instead created a fairly large set of plausible features, then used machine learning based techniques to determine which features were actually most predictive when used in a statistical model. By using this scheme our design knowledge and intuition need not directly produce a highly predictive set of features, but instead need only *include* high quality features along with any number of less useful features. This also allows us to speculatively try a range of features, many of which may not bear fruit, but some of which may prove unexpectedly useful.

To employ this approach, we needed a large dataset of recorded user actions that had been accurately labeled as actions taken by skilled or novice users *i.e.*, a body of *labeled training data*. In the remainder of this section we detail the collection of this data including experimental manipulations to work novice users through a learning curve so that both novice and skilled usage data could be collected. We then discuss several forms of validation of this manipulation. Following that, we turn to the generation and selection of features and the construction of a predictive model. Finally, we consider validation of the resulting model.

Application Setting for Data Collection

To produce a predictive model that will reflect real world user behavior it is important to collect data from realistic user actions. At the same time, to collect large numbers of high quality labels, accurate and controlled knowledge of expertise level for each action is needed. This is information that users may not have, and would find burdensome to report frequently (*e.g.* after every menu action). Our solution was to have users perform realistic activities in a real world application, but in a laboratory setting where we could ensure that users passed through a learning curve to provide both novice and skilled data for similar actions (at different points in time).

To accomplish this we chose to use a slightly modified version of an image editing program. Specifically, we used the GNU Image Manipulation Program (GIMP) 2.2.8 (http://www.gimp.org), which is built with GTK+ 2.6.10 (http://www.gtk.org), for our data collection. Both the GIMP and GTK are open source, allowing us to easily make modifications to facilitate our data collection.

First, menu items were added for the user to start each task or trial, and "close" and "quit" were removed so the participant could not accidentally stop the study. In case a participant made serious errors in completing an assigned task, we saved the participant's work after each trial (each trial was dependent on the output of the previous one).

All actions in the GIMP could be accomplished through popup menus, which are organized much like those of Adobe Photoshop. Menu bars and toolboxes were removed so participants could only use the GIMP's popup menus, accessible with a right click. This allowed us to be assured of a high density of menu selections so that data collection would proceed quickly.

All mouse events were logged using a modified version of XNEE 2.00 (http://www.gnu.org/software/xnee/), which received data directly from the X11 windowing system. Although menu information could also be gathered this way, for speed of implementation, we chose to insert code into the GTK to log the appearance and disappearance of all menus and submenus, all menu selections and deselections (mouse entered or left a menu item) and all mouse button interactions with the menus. We were careful not to record any information that is difficult to gather in an application independent way from today's operating systems (such as the specific width and height of a menu item).

- Open the *Layers Dialog* and make sure **Top Layer** is selected
    - **Close** the layers dialog
- Make an *Ellipse (circular) Selection* in the upper left corner.
- Open the *Colors Dialog* and select a **light (pastel) pink.**
    - **Close** the colors dialog
- *Fill* that selection pink
- Make all the color in that selection in the layer *Transparent*.
- Add a *Waves filter* to the selection.

Figure 2. Instructions for one of the two tasks users completed. Note that the names of menu items and important actions are visibly salient.

Detecting Informative Moments

We define *informative moments* as user actions (or portions of user actions) which can be readily isolated, are indicative of the phenomena we wish to study, model or predict, and can be easily and accurately labeled. Detecting informative moments can be difficult because accurately detecting their start and end points may require knowledge about the user's goals. However, there are some goal-independent informative moments that can be easily identified. For our problem a very common interaction technique – specifically menu selection – serves these needs.

To this end we segmented the data that we gathered into informative moments consisting of menu operations (starting from the right click to open a pop-up menu and ending with the left click to select a menu item or dismiss the menu without a selection). Note that as a part of our development process we also considered feature extraction from other potential informative moments and from more general and undifferentiated input data. However, menu use is one of the most ubiquitous interactions in current interfaces, and use of these other features did not improve our classifier's performance, so we do not discuss them further.

Method and Participants

Participants were very briefly introduced to the GIMP and shown how to access menus with the right mouse button. They were then asked to complete two separate tasks in a fixed order. Each task consisted of seven identical trials. Ten menu selections were required for each trial. This experiment was designed to be repetitive so that participants would progress from novice to skilled behavior. As confirmed by the validation presented later, at the end of the study, the novices had learned the details of their tasks and we were able to label their actions as skilled performance.

Participants were given specific sequential instructions for each task on paper, which they could write on to keep track of their progress. These instructions were formatted so the goal of each step would be clear (Figure 2). In the first task participants drew transparent shapes and changed the background pattern on the canvas for each trial. Figure 1 illustrates the typical state of the interface after the 7th trial for this task. In the second task they drew letters and shapes and colored them with solid colors or gradients.

Participants were paid $15 for their participation in the study, which usually lasted about 90 minutes. The experimenter observed participants as they worked on the tasks, and only intervened if the participant spent two minutes searching for a menu item or if they skipped a step. The experimenter then told the participant where to find the menu item.

We collected data from 44 participants (19 female) whose average age was 25.3 years (SD = 8.37, Min = 19, Max = 59). All but two used Windows as their primary operating system. The majority of these participants reported that they were novice users of image editing and drawing manipulation programs. For example, the average participant used Microsoft Paint 2-3 times per month. None had used the GIMP before.

To help verify the novice status of each participant, they each completed a questionnaire about their experience using image editing and drawing manipulation applications before completing the tasks. Participants also answered specific questions about the location of menu items with fill in the blank questions. For example: To open a new file, I need to select the "______" menu option in the "_______" menu (where the first blank is "new" and the second blank is "file"). We instructed users to leave a question blank if they did not know the answer.

We found that most participants did not know the location of most menu items, particularly those located in sub menus of multi-level menus However, all but four participants knew that to undo an action, they should select "undo" from the "edit" submenu. Nearly all users knew the location of menu items like "undo," and would likely act at a high level of skill when selecting such items. To simplify labeling, we removed all "undo" menu selections from our analysis. No other menu items that we tested users' knowledge of had this problem, and our tasks did not include other commonly used menu items such as copy or paste. We also omitted menu selections that started a trial as detailed timing on these actions may have been confounded with task startup effects.

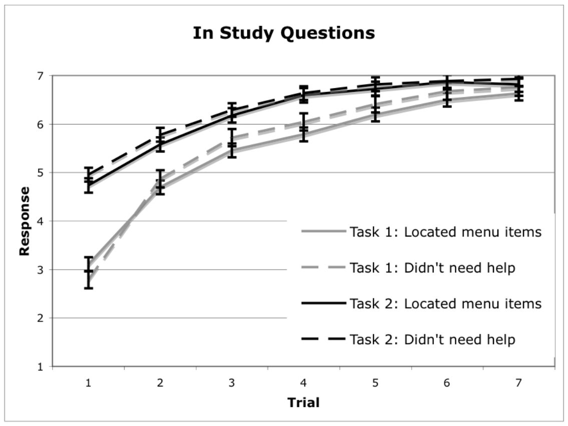


Figure 3. Plot of average of participant's subjective responses to questions asked after each trial: #1 "I had no problem locating the menu items in this trial" #2 "It was easy for me to complete this trial without external help."

To confirm that poor reading speed would not confound results, participants were asked to read a 153 word passage and reading speed was recorded. Participant's reading speed ranged from 230 words per minute (wpm) to 612 wpm (M = 349, SD = 91). All but 2 were above the adult average of 250-300 wpm [2].

We also systematically removed extreme outliers from our training data set. Specifically we calculated the Mahalanobis Distance [19] for all participants using all 46 features that we developed, and omitted training data from four participants whose average Mahalanobis distance was more than two standard deviations away from the mean. Possible reasons these participants were outliers included that one of them normally wears glasses but didn't bring them to the study, and that one was a very inexperienced computer user. The other two participants had difficultly staying on task and would often skip sections of the trials even when the experimenter encouraged them not to. In addition, we were also forced to eliminate data from three participants whose data logs were incomplete due to technical failures. This left us with data from 37 participants.

Labeling and Validating Novice vs. Skilled Behavior

Learning happened quickly and followed the power law of practice (Figure 4). Participants confirmed their subjective feelings of expertise level in a short questionnaire after each trial.

We labeled actions in a user's first trial of the first task as "novice." We labeled actions from a user's final trial in both tasks as "skilled." However, because our investigations showed that the first trial of the second task was not consistently similar to either novice or skilled behavior (performance was close to novice, but fell between these extremes) we did not label the first task of the second trial as "novice"

We collected roughly 600 examples of menu searches that were labeled as novice, and 700 that were labeled as skilled use. While perfect execution of the tasks would have resulted in twice as many examples for skilled cases compared to novice, our sample is closer to balanced because the novice examples included notably more false steps (*e.g.*, menu selections that were subsequently undone).

After completing each trial, participants were asked two questions that related to their performance to gauge their subjective impression of their own expertise on a scale of 1 to 7: "I had no problem locating the menu items in this trial" and "It was easy for me to complete this trial without external help." Figure 3 shows mean responses for these questions and clearly shows the perception of a learning effect.

As a more objective measure we can also compare performance times for menu selections within various trials with those predicted by a Keystroke Level Model (which should approximate the performance expected of a skilled user). We divided this analysis into groups defined by the submenu depth of the menu item selected (since KLM pre-

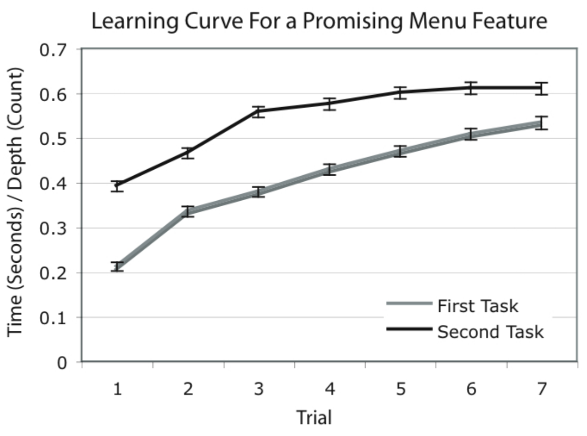


Figure 4. Plot of a promising menu feature's mean for each trial number. Note the rise in the learning curve between the first and second tasks.

dictions differ with changes in submenu depth). Expertise develops more quickly for top level menus than 2nd level submenus, and more quickly for 2nd level submenus than 3rd level and so on, because the frequency with which users encounter deeper submenus is lower. Additionally, our tasks depended most often on 2nd level submenu items.

Our analysis showed that users progressed through a learning curve. We found that, on average, when visiting 2nd level submenus users were performing at or better than the KLM predicted time by the fourth trial in the first task, and by the first trial in the second task. Users reached the KLM predicted time for 3rd level submenus by the end of the second task. This indicates that the performance reached by the end of the study was equivalent to that of skilled users.

To further explore the details of the learning curve we selected one of our most promising quantitative features – the ratio of time to make a menu selection vs. the depth of the selection (see the detailed discussion of this feature in the next section) – and considered how it varied across trials. The mean of this value across trails is displayed in Figure 4. Here we can again see a curve characteristic of a learning effect. In addition, it would appear that learning continues across tasks with the beginning of the second task picking up near the end of the first.

As illustrated in Figure 4 and verified by a two-way RM-ANOVA participant performance improved significantly over time, with main effects for Task and Trial (Wilks Λ=.379, F(1, 27) = 44.303, p<.001, multivariate η^2 = .621 and Wilks Λ=.071, F(1, 22) = 47.757, p<.001, multivariate η^2 = .929, respectively). Follow-up contrasts were significant, F(1, 27) = 233.416, p<.001. Pairwise comparisons showed significant differences between all trials but 3 and 4, and 5, 6, and 7, with means increasing over time. This suggests that learning began to tail off towards the end of each task. To confirm this, we fitted the mean values over each trial to the log of both axes to check if the curve conformed to the power law of practice [21]. The correlation was .99 for task one, t(6) = 350.87, p=.0001, and the correlation was .96 for task two, t(6) = 128.38, p=.0001. It follows that the learning curve for the second task would be less steep than that for the first task because learning occurs across tasks as well as within tasks.

Candidate Features

For each extracted informative moment we computed a set of 46 potentially predictive features. These features ranged from the amount of time spent in different parts of the menu system to characteristics of the underlying movement of the pointing device. As described previously, we generated this set of features from an understanding of human motion as described in the literature. In addition, we added candidate features based on our observations of participants as they completed the tasks. Below are the definitions of some of our more important features, organized by category.

Features derived from low-level motion characteristics

Total Time (seconds) Elapsed time within the action (starting when the menu opened and ending when it closed). (Range: 0.504 – 143)

X and Y Mouse Velocity (pixels/second) Average velocity of the mouse during a menu operation in the X and Y directions. (Range: X: 24756 – 35745; Y: 30116 – 37789)

X and Y Mouse Acceleration (change in velocity/second) Average unsigned acceleration of the mouse during a menu operation in the X and Y directions. (Range: X:0 – 242041107; Y: 0 – 1770018051.8)

Dwell Time (seconds) Time spent dwelling (not moving) during the interaction sequence. (Range: 0 – 112)

Features related to the interaction technique

Average Dwell Time (seconds/count) Time spent dwelling divided by the number of menu items visited. (Range: 0 – 3.581)

Number of Opened Submenus (count) Total number of submenus that the user opened while searching. (Range: 0 – 59)

Selection Depth (count) Depth of the selection (Range: 0 – 3) Note that this feature would not likely be predictive by itself. However, it may be useful when combined conditionally with other features.

Menu Item Visits (count) Total number of menu items that were visited or passed through during menu action. (Range: 0 – 160)

Unique Item Visits (count) Number of unique menu items visited. (Range: 1 – 57)

Selected Item Dwell Time (seconds) Time spent dwelling within the menu item that was ultimately selected. This feature sums all times spent in that item. (Range: 0 – 22)

Features related to performance models

KLM Diff (seconds) Difference between KLM predicted time and actual time for the action. (Range: 0.54 – 143.196)

KLM Ratio (dimensionless) KLM predicted time divided by the actual time for the action. (Range: 0.003 – 3.488)

Time Depth Ratio (seconds/depth) Time to make a menu selection divided by the depth of that selection. (Range: 0 – 1.368)

Feature Selection

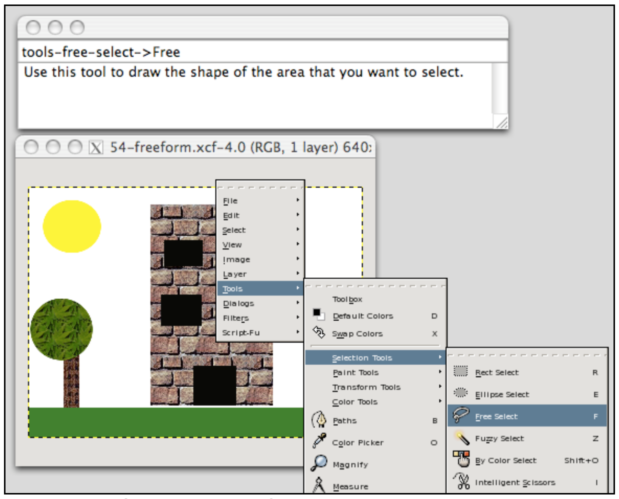
Not all features necessarily contribute to creating a good classifier. Rather than trying to predict which features would work best (which is very hard to do), an analysis of information gain [20] can be used to rank the information content of each feature (in isolation). This represents an objective estimation of how valuable each feature may be in constructing a classifier.

The following are the top 10 features ordered by the information gain ranking: (1) Average Y Acceleration, (2) KLM Diff, (3) Time Depth Ratio, (4) KLM Ratio, (5) Total Time, (6) Dwell Time, (7) Average Dwell Time, (8) Selected Item Dwell Time, (9) Menu Item Visits, (10) Number of Opened Submenus.

Note that the information gain statistic treats each feature in isolation. However, two features may contain much of the same information and so using both of them in the same classifier may not be very useful. For example, since KLM Diff and KLM Ratio are computed from the same underlying data, it is unlikely that one of them provides information that the other lacks, hence it is unlikely that using both of them would perform significantly better than just one of them. In addition, the information gain statistic does not take into account the particular properties of a particular learning algorithm.

To select features that are finely tuned to a particular learning algorithm, while taking into account any information overlap, we employ a *wrapper-based* feature selection approach [18]. This approach performs a combinatoric optimization which seeks to find the subset of possible features which produces the best accuracy result for a given learning algorithm. This technique is called wrapper-based because it can be "wrapped around" any existing learning algorithm. Although this technique is typically computationally expensive – it creates and evaluates a very large number of different classifiers – it tends to do a very good job of feature selection.

As described in the next section we eventually settled on using a decision tree classifier. A feature selection wrapper employing a genetic search optimization approach was used with this learning algorithm to select the following features: Average Y Acceleration, Menu Item Visits, Selection Depth, Time Depth Ratio, and Number of Unique Visits. These five features were used to build the statistical model described next.


tools-free-select->Free

Use this tool to draw the shape of the area that you want to select.

Figure 5. Screenshot of prototype application that gives user tailored help information about the currently highlighted menu, based on automatic classification of user expertise.

Building and Validation of a Classifier

Using the features described above, a classifier was built using the C4.5 Decision Tree learning algorithm [24] as implemented in the WEKA machine learning environment [27]. Note that we also considered several other learning algorithms including Bayesian Networks, Naïve Bayes, Support Vector Machines, and Linear Discriminant Analysis (along with the corresponding feature selection wrappers). However, Decision Trees consistently gave us the best results.

To test the effectiveness of our classifier we employed a traditional 10 fold cross-validation test. In this test a random 10% of the data is removed (or *held-out*) to form a test set. A classifier is built using the remaining 90% of the data. The accuracy of the classifier is the tested on predicting the 10% hold-out set. This process is performed 10 times with 10 disjoint hold-out test sets and the average accuracy across these 10 trials is reported. Using this validation measure our classifier achieved an accuracy of 91%.

Since the decision tree was generated automatically, it is not easily readable. However it can be described qualitatively in terms of the trends it looked for to make a classification. Characteristics of menu selections that were labeled as "novice" generally had a low average Y acceleration (meaning the mouse moved slowly, stopped, or changed direction) took longer to make the selection for a given submenu depth, and had large total number of menu selections and unique menu selections (indicating exploration of submenus). Characteristics of menu selections the classifier labeled as "skilled" had a high average Y acceleration (indicating an increase in velocity and overall speed), faster navigation of deeper menu items, and low total numbers of menu selections (indicating little exploration of additional submenus).

The result of the study, feature analysis, and training described above was a working classifier that had been both tested and trained on data from our controlled study. To close the loop, we next implemented an adaptive system that could leverage that classifier to provide tailored support to users and tested it with both a scripted and free-form task.

GOING LIVE: AUTOMATICALLY DETECTING AND CLASSIFYING EXPERTISE

Interfaces that can automatically sense and/or adapt to user expertise offer many potential benefits including just in time information, personalized help, and performance optimizations. Past work in this domain can be characterized by one of three goals: support for novice use, helping novices become skilled users, and adaptations to support skilled use.

Applications that focus strictly on support for novice behavior are designed for users who interact with an application so infrequently that they have little interest or opportunity to become a skilled user. Supporting novice use tends to focus on visual information such as intelligent previews, roadmaps showing application organization, or information about historic interactions. Novices may be willing to trade small sacrifices in task performance for substantially better affordances, feedback, and other forms of visible information.

One successful way to increase expertise is to give answers about why something didn't work [14], or more generally to increase their knowledge by explaining system behavior. The path to expertise could also be enhanced with additional information about advanced features of the interface or performance optimizations such as keyboard shortcuts.

Skilled users are likely to benefit most from support that optimizes performance aspects of an application. Split Menus [25], which move the most frequently used menu items to the top of a menu, are an example of this approach. Skilled users could also benefit from information about advanced features of the interface or performance optimizations such as keyboard shortcuts.

Prototype Application to Adapt to Expertise

To begin to explore the use of our classifier in real applications we developed a simple adaptive tool that can function across any GTK application. Our tool, which consists of a single on-screen window (top, Figure 5), displays the name of, and an expertise-tailored description of, the menu item that is currently highlighted, depending on the classification provided by our trained statistical models.

Descriptions for novices include examples of what would happen if the menu item were selected and attempt to minimize the use of domain specific knowledge, information shown to improve novice performance [7]. . Descriptions for users displaying skilled behavior include alternative names for different functions, and also include the keyboard shortcuts for different menu items. For example a description of the freehand selection tool for a novice said "Use this tool to draw the shape of the area that you want to select" compared to a more complicated description "Lasso: use this tool to make a free-hand selection. Either close a selection by ending in the point you started from, or let the tool automatically close an open shape." The set of descriptions can be configured by providing a data file with alternate text for each classification associated with each menu item.

We implemented a real-time "live" classifier in Java for parsing data about mouse and menu interactions and retrieving a prediction about whether an action was skilled or novice. Data was gathered using the same tools developed for our study (GTK and XNEE). To reduce "jitter", we aggregated the predictions using an exponentially decaying average (with a decay factor of 0.5). Our classifier reports the aggregate prediction to the adaptive tool, which monitors the currently highlighted menu item and selects the appropriate description of that menu item to display.

Validating Ability to Detect Expertise

A preliminary evaluation of the "live" classifier and its use for adaptation was conducted with 4 participants (2 female, ages = 21, 24, 32, 34). Participants were paid $15 for a 90 minute study where they performed a scripted repetitive task as well as a free-form one using our modified GIMP application. To familiarize themselves with the GIMP, participants first completed the scripted task, which was the same task used in the data collection study, and were given 45 minutes to work on it. Next participants completed a free-form task where they were told to draw a scene, and were given 30 minutes to complete this task. They worked on this free-form task for 30 minutes, of which they used the adaptation for the last 15.

Figure 6 illustrates the moving average (window size = 10) for the reported expertise predictions. Each participant's transition between the two tasks is marked with a vertical line. The classifier correctly identified that all participants started as novices (although only briefly for participant 3), and exhibited skilled behavior during the study. However, the four participants had different learning curves and chose different strategies for the free-form task. Participants 1 and 2 mostly used menu items from the scripted task in the free-from section. However, participants 3 and 4 chose to explore the menus at the start of the free-form task. As a result, they returned to a novice state and then worked back up to skilled performance.

Observationally we noted that participant 4 had significant difficulties with the scripted task and in fact this person only completed two trials of the task. This is reflected in the fluctuations in performance leading to participant 4's moving average rising and falling multiple times. Also, the maximum of the moving average for participant 4 stayed well below the expertise level achieved by other participants for almost the entire time.

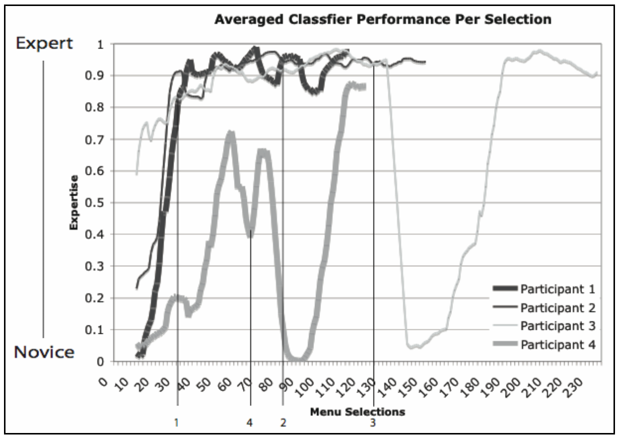


Figure 6. Moving average of live classifier predictions for repetitive and free-form tasks. The vertical bars indicate the transition between tasks.

In a post study interview, all participants said that they found the example descriptions in the adaptive help system extremely useful. They all responded positively towards applications being more aware of their activities and needs, and had no reservations about automatic evaluations of their performance. However, they expressed concern about having the ability to control the adaptation if it was based on an incorrect classification or if it hindered their activity.

In summary, although our sample was small, this study provides proof-of-concept support for our ability to close the loop and provide live adaptations based on classifications of expertise. Additionally, although we cannot draw quantitative conclusions about real world accuracy, our participants represent a range of usage styles and abilities, and our live classifier appropriately reflected this range.

CONCLUSIONS AND FUTURE WORK

Novices and skilled users differ, and these differences are often ignored by applications. Our work shows that these differences can be easily sensed with high accuracy (91%) using only a few key features. We argue that this information could be used to better adapt to user needs.

Although we have not validated the technique across multiple applications, the features we developed could be used in any application because they are not application specific. We are currently working to extend our software so it will work with more commonly used applications and the Windows Operating System. We also plan to explore how our models perform in a wider range of real world situations. This work is a key step towards a larger goal of being able to detect and understand user needs so that user interfaces can be automatically adapted to fit those needs.

ACKNOWLEDGMENTS

The authors would like to acknowledge the contributions of Jenny Hwang, Jim Lin, Aubrey Shick, and Daniel Zinzow for their work on this project. This work was supported by NSF IIS-0511895, IIS 0121560, IIS 0325351, and the first author's NSF Graduate Research Fellowship.

REFERENCES

1. Accot, J., Zhai, S. (1997). Beyond Fitts' law: models for trajectory-based HCI tasks. *Proc. CHI 1997*, ACM Press: 295-302.
2. Akoumianakis, D., Savidis, A.,, Stephanidis, C. (2000). Encapsulating intelligent interactive behavior in unified user interface artifacts. *Interacting with Computers*, 12: 383-408.
3. Bailey, B., (2000). UI Design Newsletter - Insights from Human Factors International: http://www.humanfactors.com/downloads/aug00.asp
4. Byrne, M. D., Anderson, J. R., Douglass, S., Matessa, M. (1999). Eye tracking the visual search of click-down menus. *Proc. CHI 1999*, ACM Press: 402-409.
5. Card, S. K. (1982). User perceptual mechanisms in the search of computer command menus. Proc. *Human Factors 1982*, ACM Press: 190-196.
6. Card, S. K., Moran, T. P., Newell A. (1983). *The Psychology of Human-Computer Interaction.* Lawrence Erlbaum Associates, Inc.
7. Dumas, S., Landauer, T. K., (1983). Using Examples to Describe Categories. *Proc CHI 1983,*ACM Press: 112-115.
8. Everett, S. P., Byrne, M. D. (2004). Unintended effects: Varying icon spacing changes users' visual search strategy. *Proc. CHI 2004*, ACM Press: 695-702.
9. Fitts, P. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47(6): 381-391.
10. Hendrickson, J. J. (1989). Performance, preference, and visual scan patterns on a menu-based system: implications for interface design. *Proc. CHI 1989*, ACM Press: 217-222.
11. Hornof, A.J., Kieras, D.E., (1999). Cognitive modeling demonstrates how people use anticipated location knowledge of menu items. *Proc. CHI 1999*, ACM Press: 410-417.
12. Hudson, S.E., Fogarty, J., Atkeson, C.G., Avrahami, D., Forlizzi, J., Kiesler, S., Lee, J.C., and Yang, J (2000). Predicting human interruptibility with sensors: A Wizard of Oz feasibility study. *Proc. CHI 2000*, ACM Press: 257-264.
13. Kaptelinin, V. (1993). Item recognition in menu selection: the effect of practice. *Proc. Interact 1993 and CHI 1993,* ACM Press: 183-184.
14. Ko, A.J. and Myers, B.A. (2004). Designing the Whyline: A debugging interface for asking questions about program failures. *Proc. CHI 2004,* ACM Press: 151-158.
15. Lane, D., Napier, H., Peres, C., and Sandor, A. (2005). The hidden costs of graphical user interfaces: The failure to make the transition from menus and icon tool bars to keyboard shortcuts. *International Journal of Human-Computer Interaction*, 18: 133-144.
16. Lieberman, H. (1998). Integrating User Interface Agents with Conventional Applications. *Knowledge-Based Systems. 11(1: 15-23.*

17. John, B. E., Kieras, D. E. (1996). The GOMS family of user interface analysis techniques: comparison and contrast. *ACM Transactions on Computer Human Interaction*. 3(4): 320-351.
18. Kohavi, R., John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, 1-2, 273-324.
19. Mahalanobis, P.C., (1930). On tests and measures of groups divergence. *Journal of the Asiatic Society of Benagal*, 26.
20. Mitchell, T.M. (1997). *Machine Learning*. McGraw-Hill.
21. Newell, A., Rosenbloom, P.S. (1981). *Mechanisms of skill acquisition and the law of practice*, in *Cognitive skills and their acquisition*, J.R. Anderson, Editor. Lawrence Erlbaum Associates: Hillsdale, NJ. p. 1-56.
22. Norman, D. (1988). *Design of Everyday Things*, Doubleday.
23. Perlman, G. (1984). Making the right choices with menus. *Proc. Interact 1984*, Elsevier Science Publishers: 317-321.
24. Quinlan, J.R. (1993). *C4.5: Programs for Machine Learning,* Morgan Kaufmann.
25. Sears, A., Shneiderman, B. (1994). Split menus: effectively using selection frequency to organize menus. ACM *Transactions on Computer Human Interaction.* 1(1): 27-51.
26. Somberg, B. L. (1987). A comparison of rule-based and positionally constant arrangements of computer menu items. *Proc. CHI/GI 1987*, ACM Press: 255-260.
27. Witten, I.H., Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques,* 2nd Edition, *Morgan Kaufmann*, San Francisco.