On Improving Application Utility Prediction

Joshua Hailpern
University of Illinois
201 N Goodwin Ave
Urbana, IL 61801
jhailpe2@cs.uiuc.edu

Joseph Subida
University of Illinois
201 N Goodwin Ave
Urbana, IL 61801
jsubida2@illinois.edu

Nicholas Jitkoff
Google
1600 Amphitheatre Pky
Authors Square
Mtn. View, CA 94043
alcor@google.com

Karrie Karahalios
University of Illinois
201 N Goodwin Ave
Urbana, IL 61801
kkarahal@cs.uiuc.edu

CHI 2010, April 10–15, 2010, Atlanta, Georgia, USA.

ACM 978-1-60558-930-5/10/04.

Abstract
When using the computer, each user has some notion that "these applications are important" at a given point in time. We term this subset of applications that the user values as **high-utility applications**. Identifying these high-utility applications is critical to the fields of Task Analysis, User Interruptions, Workflow Analysis, and Goal Prediction. Yet, existing techniques to identify high-utility applications are based upon task identification, conglomeration of related windows, limited qualitative observation, or common sense. Our work directly associates measurable computer interaction (CPU consumption, window area, etc.) with the user's perceived application utility. In this paper, we present an objective utility function that accurately predicts the user's subjective impressions of application importance. Our work is based upon 321 hours of real-world data from 22 users (both professional and academic) improving existing techniques by over 53%.

Keywords
Application Utility, Application Importance, Modeling

ACM Classification Keywords
H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

General Terms
Experimentation, Human Factors

Introduction

Applications are the gateway to computer usage. The nature of most modern computer systems (high CPU, RAM, screen real-estate, and other system resource availability) allows many concurrent applications to be open at once, regardless of whether a given application is being actively used or not. Consider a graphic designer working on a large client presentation. This user might have a presentation software open and in focus, while also busily entering text. Meanwhile, the designer might also have a PDF viewer open for reference (not in focus but mostly visible), which the user glances at periodically. Though partially obscured, a web browser pointing at a social networking site, might also sit on the desktop, while a media player plays music in the background. If asked to recall which of these applications were related to presentation design, the user could easily state "presentation software and PDF viewer." Yet how can a computer determine whether an application is related/relevant to the current work? How can the utility of an application be gauged?

While a computer has difficulty predicting relevant applications, each user can recall "these applications were important" at a given point in time. We term the subset of applications the user values as **high-utility applications**. This paper aims to bridge this gap by bringing together a user's perceived application utility and the system's measured resource allocation. Consider the scenario above: while the presentation software might play a central role for this designer, the PDF viewer might also be relevant. The media player is not relevant, though it is constantly active. In other words, the contribution of this work is a *general utility function that accurately predicts a user's impression of application utility (based on a large and diverse data set)*.

Background

For years, computer scientists have indirectly attempted to automatically determine the applications that are "important" to a user through Task Analysis [6,12] and Time Management/Workflow analysis [1,4]. This research, and the research in the field of Interruption Detection [8], suggest there is a quantifiable link between system resource utilization and application utility. It is this link that we aim to further explore, to better quantify, and to accurately predict. To date, existing approaches have only examined a small subset of interaction metrics to predict task or workflow, with minimal evidence to justify their variable selection. Work by [7] suggest that to truly understand users we must examine behavior from a wide spectrum of UI interactions, including both short- and long-term use patterns. Moreover, [9] and [10] suggest we cannot fully translate complex interaction behavior into a series of steps, or tasks, towards an end goal. While current research has attempted to identify tasks as a conglomeration of windows that might be related, we have taken a different approach. Our research has focused on directly quantifying an application's utility to the user at a given point in time.

Research Question

We introduce the following research question that will guide the remainder of this paper:

> *Is there a general utility function based on measurable aspects of computer usage (e.g., CPU, window size, etc.) that accurately predicts the user's impression of application utility?*

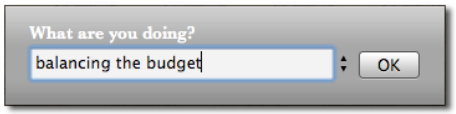

Figure 1. Current Activity Prompt
Users would be alerted by brief sound, and dimming of entire screen. Prompt window would then appear in center of the screen.

Figure 2. User Debriefing Window
Users are presented their recorded activity, date, time and all active applications. Users select all related applications (in this example Calculator and Quicken 2007 are selected).

In other words, we aim to link actual system-level activity with perceived application utility to create a predictive model of application value. To achieve this end, we conducted a experience sampling study, whose data was used to build predictive models.

Methods

To examine our research question and accurately model computer usage, we recruited 36 users to participate in a week-long (5 day), real-world, data collection process. Of the 36 users, 22 (61%) agreed to participate, completed the process, and returned data. The 22 participants were 55% male, with an age range of 21 to 59 (median 46) recruited from 6 companies and 5 universities with educational backgrounds in more than 17 different areas. Our goal was to link artifacts of computer usage (e.g., CPU, window size, etc.) with perceived application utility at a given point in time. To achieve this, we designed, built, and distributed the CLOTHO (**C**omputer **L**ogistical **O**perations and **T**emporal **H**uman **O**bservation) system. CLOTHO allowed us to collect computer resource allocation and UI interactions (*predictive variables*), and link them with human generated data (*dependent variable*). Using CLOTHO, we collected 321 hours of computer and human data over a total of 126 user-days (resulting in 2,294 sets of data points). CLOTHO was built in Cocoa and run on MACs with OS X 10.5+.

Data Collection (Predictive Variables)
Throughout the day, CLOTHO would collect 11,899 different metrics of computer usage ranging from CPU and RAM consumption to visible area of individual windows. We hypothesized that these *predictive variables* could be used to create a predictive model of application utility.

Data Collection (Dependent Variable)
To capture perceived application utility over the experimental period, we utilized recall-based experience sampling data collection [3]. Memory, or recall, is often more important than reality [11]. Therefore, we believe that modeling user's recalled application utility would provide a more usable model for real-world applications. CLOTHO would periodically prompt (Figure 1) users for their current activity or goal. To ground the prompt interval selections in existing literature, timings were based on average, coarse, and medium breakpoint intervals in [8]. Conceptually, these breakpoints occur at conceptual shifts in workflow (changing activity and switching documents, respectively), providing us with a large set of distinct activities.

At the end of each day, users ran a debriefing program which randomly selected 20 activities from the past day. A description was presented to the user along with a time-stamp and an icon grid of all open applications at that time (Figure 2). Participants were asked to select all applications (binary for each application: yes/no) that were directly related to the specified activity and time by clicking the icon/name. This provided us with our *dependent variable*.

Analytical & Statistical Methods

Given the large feature set associated with each dependent variable, we modeled application utility using machine learning, similar to the approach of Brugge et al., and Dragunov et al.[1,5] but with a different end goal and data sets. Our primary data set consisted of debriefed data from the "past 24 hours." Thus, our models were trained and tested on 16,591 data points (whether an application was, or was not,

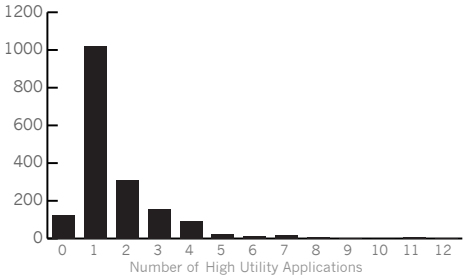

Figure 3. Histogram of Number of High Utility Applications
note similarity to Pareto distribution

part of the specified activity) each of which was associated with 11,899 predictive variables. 3,002 data points were marked as high utility applications (18.09%)

We examined three modeling approaches to create a utility function: Naive Bayes, Logistic Regression, and Decision Tree. We trained our model on 80% of the data points (randomly selected), holding 20% in reserve for accuracy testing. To ensure robustness of the resulting predictive accuracy, we performed a 5-way cross validation. Due to the high percentage of low-utility applications (81.91%), we also created a naive model whose strategy was always guessing "low-utility."

We compared the four models against the current practice of associating window focus with application utility. Before any complex analysis we can see an inherent flaw in this approach. The mean number of high utility applications across all users was 1.69. When examined as a set of tabulated frequencies (Figure 3), it is apparent that 35.62% of the time users had more than one high utility application. In other words, using a binary feature to determine application utility (e.g., which application has focus) will fail to predict all the high utility applications 35.62% of the time (assuming that feature is 100% accurate) because only one application can have focus at any given moment.

Results

The technical definition of a model's Accuracy score is percent agree / total. Our *Decision Tree* has the highest value (89.13%) followed by *Application Focus* (87.06%) and *Logistic Regression* (86.76%). Naive Bayes also has a high score (82.66%), almost equal to *Always Guess 0*, which blindly ranks all applications as low utility (81.87%). Although each of these models have a high Accuracy score, *Always Guessing 0* (which is useless in practice) is equally high. This is a direct result of the large percentage of low utility applications. **The Accuracy scores are greatly inflated, rendering them useless as a measure of each model's ability to predict high utility applications — the most relevant feature to the user.** This is a known issue with using accuracy as a metric of model quality in data mining and information retrieval [2].

Consider a Monty Hall-type problem with 100 doors and only one prize: If you guess all doors to have no prize, you will have a 99% accuracy for guessing what a door contains. This results in a high Accuracy score (only 1% away from 100% accuracy) yet you will almost always lose the game. In other words, 99% accuracy is not always "good," depending on the context. Similarly, if we are identifying high-utility applications, having a 82% accuracy by always guessing 0 (low-utility) will have a high accuracy score, yet will be an ineffective model since the real strength of these predictive models lie in correctly identifying the high utility applications. Thus, the Accuracy score likely obscures the true agreement between the predictive and perceived application utility of the model.

Cohen's kappa is a statistical measure of accuracy (giving weight to positives and negatives), and presenting a measure more in tune with the behavior of a model against an accepted standard. Kappa also takes into account events agreeing by chance (whereas Accuracy does not). Table 1 illustrates the great change between the current approach (*Utility is Application Focus*) and the *Decision Tree* model. Moreover

	Kappa Score	Sensitivity	Specificity	Recall	Precision
Always Guess 0	0.00 [na]	0.00	1.00	0.00	0.00
Utility is App Focus	0.48 [0.44, 0.52]	0.43	0.97	0.75	0.43
Naive Bayes	0.39 [0.35, 0.43]	0.47	0.90	0.52	0.47
Logistic Regression	0.53 [0.49, 0.57]	0.58	0.93	0.65	0.58
Decision Tree	0.62 [0.58, 0.65]	0.66	0.95	0.72	0.66

Table 1. Statistical Analysis of Models
kappa score includes confidence intervals for 95%

improvements between Decision Tree, and all other models (including Window Focus) are all highly significant ($p<0.01$). In other words, Decision Tree has the highest agreement with the original data, and this improvement is statistically significant compared to Naive Bayes, Logistic Regression, Always Guess 0, and Utility is Application Focus.

Perhaps the most representative measure of the model's ability to predict high utility applications is through a sensitivity analysis. Sensitivity is the proportion of high-utility applications that were correctly identified as such. Decision Tree had a sensitivity of 66%. This improves on the sensitivity of Application Focus by 53%. In other words Decision Tree is significantly more sensitive than the currently approach, using Application Focus.

Discussion

Our results show that with a relatively low-cost Decision Tree model, we can build a accurate application utility function (65% of actual user-specified high-utility applications are predicted as being high-utility by the model). More importantly, this is a 53% increase over the current method for predicting application utility. In other words, the accuracy of the generated predictive models demonstrates a strong potential for computerized systems to accurately predict high-utility applications.

With application utility being a central feature for so many other disciplines, a light-weight Decision Tree model can greatly increase the prediction of high utility applications, and therefore increase the ability of interruption systems to detect breakpoints or task analysis systems to associate relevant applications. Further, AI systems which wish to provide assistance to users' workflow by identifying which applications they are using, can utilize our model to far more accurately detect the high utility applications in use.

CONCLUSION

In this paper, we have examined the interrelationship between system resource consumption, and perceived application utility by users. From a large data set of 22 users and 321 real-world hours of data, we constructed a set of models for predicting application utility. The most accurate model created improved upon existing techniques by 53% resulting in a Decision Tree with a 66% accuracy for predicting actual high-utility applications as high-utility, and a "good" kappa score of 0.62. This model performed statistically above the existing approach for determining high-utility applications (window focus) ($p<0.01$). Our findings further detail the degree of high-utility applications that

go undetected through traditional measures. By examining a data set from a wide age, background, and occupation of users, we believe our findings to have broad applications across multiple domains.

We believe this work addresses a fundamental challenge for system designers that wish to provide support to users based upon the applications they are currently using. While this research is not definitive, and allows for many avenues for future investigation, we believe this work takes important first steps towards providing a more comprehensive understanding of user behavior and interaction with computer systems.

References

[1] Bruegge, B. *et al*. Classification of tasks using machine learning. *In Proc. PROMISE '09 Conference* (2009). ACM, New York, NY.

[2] Manning, C. D., Raghavan, P., and Schuetze, H. *Introduction to Information Retrieval.* Cambridge University Press, Cambridge, England, 2009.

[3] Consolvo, S. and Walker, M. Using the Experience Sampling Method to Evaluate Ubicomp Applications. *IEEE Pervasive Computing,* 2(2), 2003, 24–31.

[4] Czerwinski, M., Horvitz, E., and Wilhite, S. A diary study of task switching and interruptions. *In Proc CHI '04 Conference* (2004). ACM, New York, NY.

[5] Dragunov, A. N. *et al*. TaskTracer: a desktop environment to support multi-tasking knowledge workers. *In Proc. IUI '05 Conference* (2005). ACM, New York, NY.

[6] Fenstermacher, K. D. and Ginsburg, M. A Lightweight Framework for Cross-Application User Monitoring. *Computer,* 35(3), 2002, 51–59.

[7] Hilbert, D. M. and Redmiles, D. F. Extracting usability information from user interface events. *ACM Computing Surveys,* 32(4), 1999, 384–421.

[8] Iqbal, S. T. and Bailey, B. P. Understanding and developing models for detecting and differentiating breakpoints during interactive tasks. *In Proc. CHI '07 Conference* (2007). ACM, New York, NY.

[9] Kaasgaard, K. and Nardi, B. A. Software design and usability: Talks with Bonnie Nardi, Jakob Nielsen, David Smith, Austin Henderson & Jed Harris, Terry Winograd and Stephanie Rosenbaum. Copenhagen Business School Pr, 2000.

[10] Lim, Y.-K. Multiple aspect based task analysis (MABTA) for user requirements gathering in highly-contextualized interactive system design. *In Proc. TAMODIA '04 Conference* (2004). ACM, New York, NY.

[11] Norman, D. A. THE WAY I SEE IT Memory is more important than actuality. *interactions,* 16(2), 2009, 24–26.

[12] Oliver, N. *et al*. SWISH: semantic analysis of window titles and switching history. *In Proc. IUI '06 Conference* (2006). ACM, New York, NY.